

Real-Time Windows Target

For Use with Real-Time Workshop®

Modeling
|

Simulation
|

Implementation
|

User's Guide
Version 2



How to Contact The MathWorks:



www.mathworks.com	Web
comp.soft-sys.matlab	Newsgroup



support@mathworks.com	Technical support
suggest@mathworks.com	Product enhancement suggestions
bugs@mathworks.com	Bug reports
doc@mathworks.com	Documentation error reports
service@mathworks.com	Order status, license renewals, passcodes
info@mathworks.com	Sales, pricing, and general information



508-647-7000	Phone
--------------	-------



508-647-7001	Fax
--------------	-----



The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098	Mail
--	------

For contact information about worldwide offices, see the MathWorks Web site.

Real-Time Windows Target User's Guide

© COPYRIGHT 1999-2002 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by or for the federal government of the United States. By accepting delivery of the Program, the government hereby agrees that this software qualifies as "commercial" computer software within the meaning of FAR Part 12.212, DFARS Part 227.7202-1, DFARS Part 227.7202-3, DFARS Part 252.227-7013, and DFARS Part 252.227-7014. The terms and conditions of The MathWorks, Inc. Software License Agreement shall pertain to the government's use and disclosure of the Program and Documentation, and shall supersede any conflicting contractual terms or conditions. If this license fails to meet the government's minimum needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to MathWorks.

MATLAB, Simulink, Stateflow, Handle Graphics, and Real-Time Workshop are registered trademarks, and TargetBox is a trademark of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Printing History:	January 1999	First printing	New for Version 1.0 (Release 11.0)
	January 2000	Second printing	Revised for Version 1.5 (Release 11.1+)
	September 2000	Third printing	Revised for Version 2.0 (Release R12)
	June 2001	Online only	Revised for Version 2.1 (Release R12.1)
	July 2002	Online only	Revised for Version 2.2 (Release 13)

Preface

Required Products	ix
MATLAB	ix
Simulink	x
Real-Time Workshop	xi
C Compiler	xi
Related Products	xii
Using This Guide	xiii
Expected Background	xiii
Organization	xiv
Conventions	xv
Terminology	xv
Typographical Conventions	xvii

Introduction

1 |

What Is the Real-Time Windows Target?	1-3
Features	1-4
Real-Time Kernel	1-4
Real-Time Application	1-5
Signal Acquisition and Analysis	1-6
Parameter Tuning	1-7

Hardware Environment	1-8
PC Compatible Computer	1-8
Input/Output Driver Support	1-8
Software Environment	1-10
Nonreal-Time Simulation	1-10
Real-Time Execution	1-10
Development Process	1-11
System Concepts	1-12
Simulink External Mode	1-12
Data Buffers and Transferring Data	1-13

Installation and Configuration

2

System Requirements	2-3
Hardware Requirements	2-3
Software Requirements	2-4
Real-Time Windows Target	2-5
Getting or Updating Your License	2-5
Installing from a CD	2-6
Installing from the Web	2-7
Files on the Your Computer	2-9
Initial Working Directory	2-11
Setting the Working Directory from Desktop Icon	2-11
Setting the Working Directory from MATLAB	2-11
Third Party C Compiler	2-12
Selecting a Default C Compiler	2-12
Real-Time Windows Target Kernel	2-14
Installing the Kernel	2-14
Uninstalling the Kernel	2-16

Testing the Installation	2-18
Running the Model rtdvp.mdl	2-18
Displaying Status Information	2-21
Detecting Excessive Sample Rates	2-22
Demo Library	2-22

Basic Procedures

3

Simulink Model	3-3
Creating a Simulink Model	3-3
Entering Simulation Parameters for Simulink	3-7
Entering Scope Parameters for Signal Tracing	3-9
Running a Nonreal-Time Simulation	3-12
Real-Time Application	3-14
Entering Simulation Parameters for Real-Time Workshop	3-14
Entering Scope Parameters for Signal Tracing	3-17
Creating a Real-Time Application	3-20
Entering Additional Scope Parameters for Signal Tracing ...	3-21
Running a Real-Time Application	3-25
Running a Real-Time Application from the MATLAB Command Line	3-28
Signal Logging to the MATLAB Workspace	3-29
Entering Scope Parameters	3-29
Entering Signal and Triggering Properties	3-31
Plotting Logged Signal Data	3-35
Signal Logging to a Disk Drive	3-37
Entering Scope Parameters	3-37
Entering Signal and Triggering Properties	3-40
Entering Data Archiving Parameters	3-43
Plotting Logged Signal Data	3-46

Parameter Tuning	3-48
Types of Parameters	3-48
Changing Model Parameters	3-49

Advanced Procedures

4

I/O Boards	4-3
I/O Board Dialog Box	4-3
ISA Bus Board	4-5
PCI Bus Board	4-6
PC/104 Board	4-6
Compact PCI Board	4-6
PCMCIA Board	4-6
 I/O Driver Blocks	 4-8
Real-Time Windows Target Library	4-8
Simulink Library	4-10
Analog Input Block	4-12
Analog Output Block	4-14
Digital Input Block	4-17
Digital Output Block	4-19
Counter Input Block	4-22
Encoder Input Block	4-25
Output Signals from an I/O Block	4-27
Variations with Channel Selection	4-29
 Using Analog I/O Drivers	 4-32
I/O Driver Characteristics	4-32
Normalized Scaling for Analog Inputs	4-33

Troubleshooting

5

Plots Not Visible in Simulink Scope Block	5-2
Compiler Error Message	5-3
Failure to Connect to Target	5-3
Sample Time Too Fast	5-4
S-Functions Using Math Functions	5-5

Supported I/O Boards

A

ISA Bus	A-2
PCMCIA Bus	A-11
PCI Bus	A-12
Compact PCI	A-14
PXI Bus	A-14
PC/104 Bus	A-15
Standard Devices	A-16

Custom I/O Driver Blocks

B

Source Code for DOS Target Drivers	B-2
Incompatibility with Win32 API Calls	B-3
Nonsupported C Functions	B-3
Supported C Functions	B-4

Preface

Required Products	ix
MATLAB	ix
Simulink	x
Real-Time Workshop	xi
C Compiler	xi
Related Productsxii
Using This Guide	xiii
Expected Background	xiii
Organization	xiv
Conventionsxv
Terminologyxv
Typographical Conventions	xvii

The Real-Time Windows Target is part of a family of software products that you use to create real-time control systems. Some of these products are required while others you use for special applications.

This chapter includes the following sections:

- **Required Products** — MATLAB[®], Simulink[®], Real-Time Workshop[®], the Real-Time Windows Target, and a C compiler
- **Related Products** — Stateflow[®], Stateflow Coder, Dials & Gauges Blockset, DSP Blockset, Virtual Reality Toolbox, and Fixed-Point Blockset
- **Using This Guide** — Suggestions for learning about the Real-Time Windows Target, finding information, and a description of the chapters
- **Conventions** — Terms that may have various meanings and text formats in this guide
- **Typographical Conventions** — Overview of the format of code, MATLAB output, and menu items.

Required Products

The Real-Time Windows Target is a self-targeting system where the host and the target computer are the same computer. You can install it on a PC-compatible computer running Microsoft Windows 98, Windows NT 4.0, Windows 2000, Windows Millennium Edition, or Windows XP.

The Real-Time Windows Target requires the following products:

- **MATLAB** — Command line interface for the Real-Time Windows Target
- **Simulink** — Environment to model physical systems and controllers using block diagrams
- **Real-Time Workshop** — Converts Simulink blocks and code from Stateflow Coder into C code
- **C Compiler** — Converts C code from Real-Time Workshop into executable code. Choose either Microsoft Visual C/C++ or Watcom C/C++

MATLAB

MATLAB provides the design and analysis tools that you use when creating Simulink block diagrams.

Note Version 2.2 of the Real-Time Windows Target requires MATLAB Version 6.5 on the Release 13 CD.

MATLAB documentation — For information on using MATLAB, see the Getting Started documentation. It explains how to work with data and how to use the functions supplied with MATLAB. For a reference describing the functions supplied with MATLAB, see the online MATLAB Function Reference.

Simulink

Simulink provides an environment where you model your physical system and controller as a block diagram. You create the block diagram by using a mouse to connect blocks and a keyboard to edit block parameters.

Nonsupported Simulink blocks — You can use the Real-Time Windows Target with most Simulink blocks including discrete-time and continuous-time systems. The Real-Time Windows Target does not support blocks that do not run in real-time and the following blocks: To Workspace and To File blocks.

Limitations with Real-Time Workshop — When you use a continuous-time system and generate code with Real-Time Workshop, you must use a fixed-step integration algorithm. C-code S-functions are supported by Real-Time Workshop. However, M-code S-functions are not supported.

Real-Time Windows Target I/O driver blocks — With the Real-Time Windows Target, you can remove the physical system model and replace it with I/O driver blocks connected to your sensors and actuators. The Real-Time Windows Target I/O library supports more than 200 boards.

Note Some of the functions on a board may not be supported by the Real-Time Windows Target. Check the MathWorks Web site for an updated list of supported boards and functions at: <http://www.mathworks.com/products/rtwt/ioboards.shtml>.

Note Version 2.2 of the Real-Time Windows Target requires Simulink Version 5.0 on the Release 13 CD.

Simulink documentation — For information on using Simulink, see the Using Simulink documentation. It explains how to connect blocks to build models and change block parameters. It also provides a reference that describes each block in the standard Simulink library.

Real-Time Workshop

Real-Time Workshop provides the utilities to convert your Simulink models into C code, and then with a third-party C compiler, compile the code into a real-time executable.

The Real-Time Windows Target is designed for maximum flexibility during rapid prototyping. This flexibility allows parameter tuning and signal tracing during a real-time run, but increases the size of the generated code. However, Real-Time Workshop has other code formats that generate the more compact code needed for embedded applications.

Note Version 2.2 of the Real-Time Windows Target requires Real-Time Workshop Version 5.0 on the Release 13 CD.

More information about Real-Time Workshop — For information on code generation, see the Real-Time Workshop documentation.

C Compiler

The C compiler creates executable code from the C code generated from Real-Time Workshop and the C-code S-functions you have created.

In addition to the products from The MathWorks, you need to install a C compiler. Real-Time Workshop and the Real-Time Windows Target support the following C compilers:

- **Microsoft Visual C/C++** — Version 2.2 of the Real-Time Windows Target requires the Professional Edition of Microsoft Visual C/C++ Version 5.0, 6.0, or 7.0. The Standard Edition does not include all of the features needed to work with the Real-Time Windows Target.
- **Watcom C/C++** — Version 2.2 of the Real-Time Windows Target requires Watcom C/C++ Version 10.6 or 11.0.

Related Products

The MathWorks provides several products that are especially relevant to the kinds of tasks you can perform with the Real-Time Windows Target.

For more information about any of these products, see either

The online documentation for that product if it is installed or if you are reading the documentation from the CD

- The MathWorks Web site, at <http://www.mathworks.com>; see the “products” section

Note The toolboxes listed below all include functions that extend the capabilities of MATLAB. The blocksets all include blocks that extend the capabilities of Simulink.

Product	Description
Dials & Gauges Blockset	Monitor signals and control simulation parameters with graphical instruments
DSP Blockset	Design and simulate DSP systems
Fixed-Point Blockset	Design and simulate fixed-point systems
Stateflow	Design and simulate event-driven systems
Stateflow Coder	Generate C code from Stateflow charts
Virtual Reality Toolbox	Create and manipulate virtual reality worlds from within MATLAB and Simulink

Using This Guide

To help you effectively read and use this guide, we have provided a brief description of the chapters and a suggested reading path.

This section includes the following topics:

- “Expected Background” on page xiii
- “Organization” on page xiv

Expected Background

To benefit from reading this book, you should be familiar with

- Using Simulink and Stateflow to create models as block diagrams, and simulating those models in Simulink
- The concepts and use of Real-Time Workshop to generate executable code

When using Real-Time Workshop and the Real-Time Windows Target, you do not need to program in C or other low-level programming languages to create and test real-time systems.

If You Are a New User — Begin with Chapter 1, “Introduction.” This chapter gives you an overview of the Real-Time Windows Target features and the development environment. Next, read and try the examples in Chapter 3, “Basic Procedures.”

If You Are an Experienced Real-Time Windows Target User — We suggest you review the sections on signal tracing and signal logging in Chapter 3, “Basic Procedures.” After you are familiar with using the Real-Time Windows Target, read how to add I/O drivers to your Simulink model in Chapter 4, “Advanced Procedures.”

Organization

The following table lists the organization of this guide.

Chapter or Appendix	Description
Introduction	Overview of the functions and features of the Real-Time Windows Target
Installation and Configuration	Procedures to install the Real-Time Windows Target on your computer
Basic Procedures	Procedures to help you become familiar with using the Real-Time Windows Target
Advanced Procedures	Procedures for using I/O drivers with the Real-Time Windows Target
Troubleshooting	Solutions to some common problems
Supported I/O Boards	List of I/O boards supported by the Real-Time Windows Target with Simulink driver blocks
Custom I/O Driver Blocks	Procedures and notes for creating your own Simulink blocks using C-code S-functions

Conventions

To help you effectively read this guide, we use some conventions. Conventions are the ways of consistently formatting the text and graphics.

This section includes the following topics:

- “Terminology” on page xv
- “Typographical Conventions” on page xvii

Terminology

The following table lists some of the terms we use in this guide.

Term	Definition
application	See <i>real-time application</i> .
build process	Process of generating C code from your Simulink model, compiling and inlining the generated code to create a <i>real-time executable</i> .
execution	Running the executable code on the target PC in real time.
executable code	See <i>real-time application</i> .
external mode	Simulink mode that uses a Simulink block diagram as a graphical user interface to a <i>real-time executable</i> . This interface provides parameter downloading and signal uploading for display using Scope blocks.
kernel	Software component that handles system interrupts and regulates time interrupts when model code is stepped.
parameter tuning	Process of changing block parameters and downloading the new values to a <i>real-time executable</i> while it is running.

Term	Definition
real-time application	Code ready to run in real time with the <i>kernel</i> .
sample rate	Inverse of the sample time given as samples per second.
sample time	Length of time, in seconds, between each interrupt that the model is stepped.
self-targeting system	A system where both the development environment and real-time environment use the same processor.
signal logging	Acquire and save signal data created during a real-time execution.
signal tracing	Acquire and display of a sequence of bursts of signal data created during real-time execution. The burst length corresponds with the time axis at the Scope block.

Typographical Conventions

This manual uses some or all of these conventions.

Item	Convention	Example
Example code	Monospace font	To assign the value 5 to A, enter <code>A = 5</code>
Function names, syntax, filenames, directory/folder names, and user input	Monospace font	The <code>cos</code> function finds the cosine of each array element. Syntax line example is <code>MLGetVar ML_var_name</code>
Buttons and keys	Boldface with book title caps	Press the Enter key.
Literal strings (in syntax descriptions in reference chapters)	Monospace bold for literals	<code>f = freqspace(n, 'whole')</code>
Mathematical expressions	<i>Italics</i> for variables Standard text font for functions, operators, and constants	This vector represents the polynomial $p = x^2 + 2x + 3$.
MATLAB output	Monospace font	MATLAB responds with <code>A =</code> <code>5</code>
Menu and dialog box titles	Boldface with book title caps	Choose the File Options menu.
New terms and for emphasis	<i>Italics</i>	An <i>array</i> is an ordered collection of information.
Omitted input arguments	(...) ellipsis denotes all of the input/output arguments from preceding syntaxes.	<code>[c, ia, ib] = union(...)</code>
String variables (from a finite list)	<i>Monospace italics</i>	<code>sysc = d2c(sysd, 'method')</code>

Introduction

What Is the Real-Time Windows Target?	1-3
Features	1-4
Real-Time Kernel	1-4
Real-Time Application	1-5
Signal Acquisition and Analysis	1-6
Parameter Tuning	1-7
Hardware Environment	1-8
PC Compatible Computer	1-8
Input/Output Driver Support	1-8
Software Environment	1-10
Nonreal-Time Simulation	1-10
Real-Time Execution	1-10
Development Process	1-11
System Concepts	1-12
Simulink External Mode	1-12
Data Buffers and Transferring Data	1-13

The Real-Time Windows Target has many features. An introduction to these features and the Real-Time Windows Target software environment will help you develop a model for working with Real-Time Windows Target.

This chapter includes the following sections:

- **What Is the Real-Time Windows Target?** — A PC solution for prototyping and testing real-time systems
- **Features** — Real-time kernel, real-time application, signal acquisition and analysis, and parameter tuning
- **Hardware Environment** — PC compatible computer and I/O support boards
- **Software Environment** — Nonreal-time simulation of Simulink models and real-time execution of applications
- **System Concepts** — Simulink external mode and data buffers

What Is the Real-Time Windows Target?

The Real-Time Windows Target is a PC solution for prototyping and testing real-time systems. It is an environment where you use a single computer as a host and target.

In this environment you use your desktop or laptop PC with MATLAB[®], Simulink[®], and Stateflow[®] (optional) to create models using Simulink blocks and Stateflow diagrams.

After creating a model and simulating it with Simulink in normal mode, you can generate executable code with Real-Time Workshop[®], Stateflow Coder (optional) and a C compiler. Then, you can run your application in real time with Simulink external mode.

Integration between Simulink external mode and the Real-Time Windows Target allows you to use your Simulink model as a graphical user interface for

- **Signal visualization** — Use the same Simulink Scope blocks that you use to visualize signals during a nonreal-time simulation, to visualize signals while running a real-time application.
- **Parameter tuning** — Use the Block Parameter dialog boxes to change parameters in your application while it is running in real time.

Typical applications for the Real-Time Windows Target include

- **Real-time control** — Create a prototype of automotive, computer peripheral, and instrumentation control systems.
- **Real-time hardware-in-the-loop simulation** — Create a prototype of controllers connected to a physical plant. For example, the physical plant could be an automotive engine. Create a prototype of a plant connected to an actual controller. For example, the prototyped plant could be an aircraft engine.
- **Education** — Teach concepts and procedures for modeling, simulating, testing real-time systems, and iterating designs.

Features

The Real-Time Windows Target software environment includes many features to help you prototype and test real-time applications.

This section includes the following topics:

- “Real-Time Kernel” on page 1-4
- “Real-Time Application” on page 1-5
- “Signal Acquisition and Analysis” on page 1-6
- “Parameter Tuning” on page 1-7

Real-Time Kernel

The Real-Time Windows Target uses a small real-time kernel to ensure the real-time application runs in real time. The real-time kernel runs at CPU ring zero (privileged or kernel mode) and uses the built-in PC clock as its primary source of time:

- **Timer interrupt** — The kernel intercepts the interrupt from the PC clock before the Windows operating system receives it. This blocks any calls to the Windows operating system. Because of this, you cannot use Win32 calls in your C-code S-function.

The kernel then uses the interrupt to trigger the execution of the compiled model. As a result, the kernel is able to give the real-time application the highest priority available.

To achieve precise sampling, the kernel reprograms the PC clock to a higher frequency. Because the PC clock is also the primary source of time for the Windows operating system, the kernel sends a timer interrupt to the operating system at the original interrupt rate.

Technically, the kernel is provided as a VxD on Windows 98, and Windows Millennium Edition and as a kernel-mode driver on Windows 2000, Windows NT, and Windows XP.

- **Scheduler** — The timer interrupt clocks a simple scheduler that runs the executable. The number of tasks is equal to the number of sampling periods in the model with multitasking mode. With single-tasking mode, there is only one task. The maximum number of tasks is 32 and faster tasks have

higher priorities than slower tasks. For example, a faster task can interrupt a slower task.

During execution, the executable stores data in buffers. Later, the data in these buffers is retrieved by the Scope block. The scheduling, data storing, data transferring, and running the executable all run at CPU ring zero.

- **Communication with hardware** — The kernel interfaces and communicates with I/O hardware using I/O driver blocks, and it checks for proper installation of the I/O board. If the board has been properly installed, the drivers allow your real-time application to run.

The Analog Input, Analog Output, Digital Input, Digital Output, Counter Input, and Encoder Input blocks call the drivers for input and output. You can choose to have a driver block use values equal to voltage, normalize values from 0 to +1, normalize values from -1 to +1, or use the raw integer values from the A/D or D/A conversion press. Drivers also run at CPU ring zero.

- **Simulink external mode** — Communication between Simulink and the real-time application is through the Simulink external mode interface module. This module talks directly to the real-time kernel, and is used to start the real-time application, change parameters, and retrieve scope data.

Real-Time Application

The real-time application runs in real time on your PC computer and has the following characteristics:

- **Compiled code** — Created from the generated C code using either a Microsoft Visual C/C++ compiler or a Watcom C/C++ compiler.
- **Relation to your Simulink model** — The executable contains a binary form of all Simulink model components, connections between blocks, time dependencies, and variables in the Simulink blocks.
- **Relation to the kernel** — The executable must be loaded and executed directly by the Real-Time Windows Target kernel. It cannot be executed without the kernel.

The kernel runs as a VxD or kernel-mode driver, intercepts timer interrupts from the PC clock, maintains clock signals for the Windows operating system, and ensures real-time execution of the real-time application. As a result, both the kernel and the real-time application run at CPU ring zero.

- **Checksum** — The Simulink model and the executable contain a checksum value. The kernel uses this checksum value to determine if the Simulink model structure, at the time of code generation, is consistent with the real-time application structure during execution. This ensures that when you change parameters during an execution, the mapping of Simulink model parameters to the correct memory locations in the real-time application is correct.

If you make structural changes to your Simulink model, the Simulink checksum value will not match the executable checksum value. You will have to rebuild your executable before you can connect it to your Simulink model.

Signal Acquisition and Analysis

Signals may be acquired, displayed, and saved by using Simulink Scope blocks and Simulink external mode. This lets you observe the behavior of your model during a simulation or your application while it runs in real time.

You can acquire signal data while running your real-time application using

- **Signal Tracing** — This is the process of acquiring and visualizing signals during a real-time run. It allows you to acquire signal data and visualize it on your computer while the executable is running.
- **Signal Logging** — This is the process for acquiring signal data during a real-time run. After the run reaches its final time or you manually stop the run, you can plot and analyze the data.

You can save (log) data to variables in the MATLAB workspace or save data to your disk drive with MAT-files.

Signal logging differs from signal tracing. With signal logging you can only look at a signal after a run is finished.

For more information, see the sections “Signal Logging to the MATLAB Workspace” on page 3-29 and “Signal Logging to a Disk Drive” on page 3-37.

Parameter Tuning

Change the parameters in your Simulink model and observe the effect of those changes during a simulation or while running an application in real time.

Simulink external mode — You use Simulink external mode to connect your Simulink block diagram to your real-time application. The block diagram becomes a graphical user interface (GUI) to that executable.

Simulink external mode allows you to change parameters by editing the block diagram while running Simulink in external mode. New parameter values are automatically transferred to the real-time application while it is running.

Changing parameters — There are different types of model parameters that you can change while running your real-time application. For example, parameters include the amplitude of a gain and the frequency of a sine wave. After you connect your real-time application to your Simulink model, you can change parameters. These parameters can be changed before or while your real-time application is running by using one of the following methods:

- **Block parameters** — Change values in the dialog boxes associated with the Simulink blocks.
- **Block parameters for masked subsystems** — Change values in user-created dialog boxes associated with a subsystem.
- **MATLAB variables** — Create MATLAB variables that represent Simulink block parameters, and then change parameter values by entering the changes through the MATLAB command line.

For more information about parameter tuning, see the section “Parameter Tuning” on page 3-48.

Hardware Environment

The hardware environment consists of a PC compatible computer and I/O boards.

This section includes the following topics:

- “PC Compatible Computer” on page 1-8
- “Input/Output Driver Support” on page 1-8

PC Compatible Computer

You can use any PC compatible computer that runs Microsoft Windows 98, Windows NT 4.0, Windows 2000, Windows Millennium Edition, or Windows XP.

Your computer can be a desktop, laptop, or notebook PC.

Input/Output Driver Support

The Real-Time Windows Target uses standard and inexpensive I/O boards for PC compatible computers. When running your models in real time, the Real-Time Windows Target captures the sampled data from one or more input channels, uses the data as inputs to your block diagram model, immediately processes the data, and sends it back to the outside world through an output channel on your I/O board.

I/O boards — The Real-Time Windows Target supports a wide range of I/O boards. The list of supported I/O boards includes ISA, PCI, and PCMCIA boards. This includes analog-to-digital (A/D), digital-to-analog (D/A), digital inputs, digital outputs, and encoder inputs. In total, over 200 I/O boards are currently supported.

For a list of supported boards, see Appendix A, “Supported I/O Boards.”

Note Some of the functions on a board may not be supported by the Real-Time Windows Target. Check the MathWorks Web site for an updated list of supported boards and functions at <http://www.mathworks.com/products/rtwt/ioboards.shtml>.

I/O driver block library — The Real-Time Windows Target provides a custom Simulink block library. The I/O driver block library contains universal drivers for supported I/O boards. These universal blocks are configured to operate with the library of supported drivers. This allows easy location of driver blocks and easy configuration of I/O boards.

You drag-and-drop a universal I/O driver block from the I/O library the same way as you would from a standard Simulink block library. And you connect an I/O driver block to your model just as you would connect any standard Simulink block.

You create a real-time application in the same way as you create any other Simulink model by using standard blocks and C-code S-functions. You can add input and output devices to your Simulink model by

- Using the I/O driver blocks from the `rtwinlib` library provided with the Real-Time Windows Target. This library contains the blocks: Analog Input, Analog Output, Digital Input, and Digital Output.
- Using I/O driver blocks from the `DOSLIB` library provided with Real-Time Workshop. Use these drivers, for Keithley Metrabyte's DAS-1600 board, as a starting point to create your own custom drivers.

The Real-Time Windows Target provides driver blocks for more than 200 I/O boards. These driver blocks connect the physical world to your real-time application:

- Sensors and actuators are connected to I/O boards
- I/O boards convert voltages to numerical values and numerical values to voltages
- Numerical values are read from or written to I/O boards by the I/O drivers

Writing your own custom I/O drivers — If you need to write your own drivers, you can use the DOS Target drivers provided with Real-Time Workshop as a starting point. See Appendix B, “Custom I/O Driver Blocks.”

Software Environment

The software environment is a place to design, build, and test an application in nonreal-time and real time.

This section includes the following topics:

- “Nonreal-Time Simulation” on page 1-10
- “Real-Time Execution” on page 1-10
- “Development Process” on page 1-11

Nonreal-Time Simulation

You create a Simulink model and use Simulink in normal mode for nonreal-time simulation on your PC computer.

Simulink model — Create block diagrams in Simulink using simple drag-and-drop operations, and then enter values for the block parameters and select a sample rate.

Nonreal-time simulation — Simulink uses a computed time vector to step your Simulink model. After the outputs are computed for a given time value, Simulink immediately repeats the computations for the next time value. This process is repeated until it reaches the stop time.

Because this computed time vector is not connected to a hardware clock, the outputs are calculated in nonreal-time as fast as your computer can run. The time to run a simulation can differ significantly from real time.

Real-Time Execution

For real-time execution on your PC computer, create a real-time application and use Simulink in external mode.

Real-time application — Real-Time Workshop, the Real-Time Windows Target, and your C compiler produce an executable that the kernel can run in real time. This real-time application uses the initial parameters available from your Simulink model at the time of code generation.

If you use continuous-time components in your model and generate code with Real-Time Workshop, you must use a fixed-step integration algorithm.

Real-time execution — The Real-Time Windows Target provides the necessary software that uses the real-time resources on your computer hardware. Based on your selected sample rate, the Real-Time Windows Target uses interrupts to step your application in real time at the proper rate. With each new interrupt, the executable computes all of the block outputs from your model.

Development Process

In the Real-Time Windows Target environment, you use your desktop PC with MATLAB, Simulink, Real-Time Workshop, and the Real-Time Windows Target to

- 1 Design a control system** — Use MATLAB and the Control System Toolbox to design and select the system coefficients for your controller.
- 2 Create a Simulink model** — Use Simulink blocks to graphically model your physical system.
- 3 Run a simulation in nonreal time** — Check the behavior of your model before you create a real-time application. For example, you can check the stability of your model.
- 4 Create a real-time application** — Real-Time Workshop generates C code from your Simulink model. A third-party C compiler compiles the C code to an executable that runs with the Real-Time Windows Target kernel.
- 5 Run an application in real time** — Your desktop PC is the target computer to run the real-time application.
- 6 Analyze and visualize signal data** — Use MATLAB functions to plot data saved to the MATLAB workspace or a disk.

System Concepts

A more detailed understanding of Real-Time Workshop and the Real-Time Windows Target can help you when creating and running your real-time applications.

This section includes the following topics:

- “Simulink External Mode” on page 1-12
- “Data Buffers and Transferring Data” on page 1-13

Simulink External Mode

External mode requires a communications interface to pass parameters external to Simulink, and on the receiving end, the same communications protocol must be used to accept new parameter values and insert them in the proper memory locations for use by the real-time application. In some Real-Time Workshop targets such as Tornado/VME targets, the communications interface uses TCP/IP protocol. In the case of the Real-Time Windows Target, the host computer also serves as the target computer. Therefore, only a virtual device driver is needed to exchange parameters between MATLAB and Simulink memory space and memory that is accessible by the real-time application.

Signal acquisition — You can capture and display signals from your real-time application while it is running. Signal data is retrieved from the real-time application and displayed in the same Simulink Scope blocks you used for simulating your model.

Parameter tuning — You can change parameters in your Simulink block diagram and have the new parameters passed automatically to the real-time application. Simulink external mode changes parameters in your real-time application while it is running in real time.

As a user of the Real-Time Windows Target, you will find that the requirements for setup are minimal. You start by enabling external mode and specifying the correct name for the MEX-file interface. Then, after you have built the real-time application, you are ready for external mode operation.

Data Buffers and Transferring Data

At each sample interval of the real-time application, Simulink stores contiguous data points in memory until filling a data buffer. Once the data buffer is filled, Simulink suspends data capture while the data is transferred back to MATLAB through Simulink external mode. Your real-time application, however, continues to run. Transfer of data is less critical than maintaining deterministic real-time updates at the selected sample interval. Therefore, data transfer runs at a lower priority in the remaining CPU time after model computations are performed while waiting for another interrupt to trigger the next model update.

Data captured within one buffer is contiguous. When a buffer of data has been transferred to Simulink, it is immediately plotted in a Simulink Scope block, or it can be saved directly to a MAT-file using the data archiving feature of the Simulink external mode.

With data archiving, each buffer of data can be saved to its own MAT-file. The MAT-filenames can be automatically incremented, allowing you to capture and automatically store many data buffers. Although points within a buffer are contiguous, the time required to transfer data back to Simulink forces an intermission for data collection until the entire buffer has been transferred and may result in lost sample points between data buffers.

Installation and Configuration

System Requirements	2-3
Hardware Requirements	2-3
Software Requirements	2-4
Real-Time Windows Target	2-5
Getting or Updating Your License	2-5
Installing from a CD	2-6
Installing from the Web	2-7
Files on the Your Computer	2-9
Initial Working Directory	2-11
Setting the Working Directory from Desktop Icon	2-11
Setting the Working Directory from MATLAB	2-11
Third Party C Compiler	2-12
Selecting a Default C Compiler	2-12
Real-Time Windows Target Kernel	2-14
Installing the Kernel	2-14
Uninstalling the Kernel	2-16
Testing the Installation	2-18
Running the Model rtdvp.mdl	2-18
Displaying Status Information	2-21
Detecting Excessive Sample Rates	2-22
Demo Library	2-22

The Real-Time Windows Target requires the installation of MATLAB, Simulink, Real-Time Workshop, a C compiler, and the Real-Time Windows Target kernel. Also, make sure you set your working directory outside of the MATLAB root directory.

This chapter includes the following sections:

- **System Requirements** — Use any PC compatible computer with MATLAB, Simulink, Real-Time Workshop, the Real-Time Windows Target, and a C compiler
- **Real-Time Windows Target** — Install from a CD or download from the Web
- **Initial Working Directory** — Select a directory outside of the MATLAB root directory
- **Third Party C Compiler** — Install Microsoft Visual C/C++ or Watcom C/C++ to convert the C code from Real-Time Workshop to a real-time application
- **Real-Time Windows Target Kernel** — Install the kernel after installing the Real-Time Windows Target
- **Testing the Installation** — Use the Simulink model `rtvdp.mdl` to test the build process and a real-time application

System Requirements

The Real-Time Windows Target requires a PC compatible computer.

This section includes the following topics:

- “Hardware Requirements” on page 2-3
- “Software Requirements” on page 2-4

Hardware Requirements

The following table lists the minimum hardware resources the Real-Time Windows Target (Version 2.2) requires on your computer.

Hardware	Description
CPU	Pentium or higher in a desktop, laptop, or compact PCI or PC104 industrial computer Note The Real-Time Windows Target does not support DEC alpha computers.
Peripherals	Hard disk drive with 16 megabytes of free space Data acquisition board Note For a list of supported boards, please see http://www.mathworks.com/products/rtwt/boards/ioboards.shtml CD-ROM drive
RAM	32 megabytes or more

When using a laptop computer, the Real-Time Windows Target is a portable environment where your computer uses PCMCIA cards to interface to real world devices.

Software Requirements

The Real-Time Windows Target (Version 2.2) has certain product prerequisites that must be met for proper installation and execution.

The following table lists the software you need to install on your computer to run the Real-Time Windows Target.

Software	Description
Operating system	On supported Windows platforms
C compiler	Microsoft Visual C/C++ (Version 5.0, 6.0, or 7.0) Professional Edition Watcom C/C++ (Version 10.6 or 11.0)
MATLAB 6.5	On the Release 13 CD. Allows installation of Simulink
Simulink 5.0	On the Release 13 CD
Real-Time Workshop 5.0	On the Release 13 CD. Allows installation of the Real-Time Windows Target
Real-Time Windows Target 2.2	On the Release 13 CD or downloaded from the Web

Real-Time Windows Target

The Real-Time Windows Target (Version 2.2) is available on CD or as a Web downloadable.

If you installed the Real-Time Windows Target (Version 1.0 or 1.5) and the kernel — You need to uninstall the kernel before you can install the Real-Time Windows Target (Version 2.2). This removes the old version of the kernel.

If you did not install the Real-Time Windows Target (Version 1.0 or 1.5) — You only need to install the Real-Time Windows Target (Version 2.2).

This section includes the following topics:

- “Getting or Updating Your License” on page 2-5
- “Uninstalling the Kernel” on page 2-16
- “Installing from a CD” on page 2-6
- “Installing from the Web” on page 2-7

Getting or Updating Your License

Before you install the Real-Time Windows Target, you must have a valid Personal License Password (PLP) for each of the products you purchased.

When you purchase a product, The MathWorks sends you a Personal License Password (PLP) in an e-mail message. If you have not received a PLP number, contact the MathWorks.

Internet <http://www.mathworks.com/mla>

Log into MATLAB Access using your last name and Access number. Follow the license links to determine your PLP number

E-mail <mailto:service@mathworks.com>. Include your license number

Telephone 508-647-7000. Ask for Customer Service

Fax 508-647-7001. Include your license number

Installing from a CD

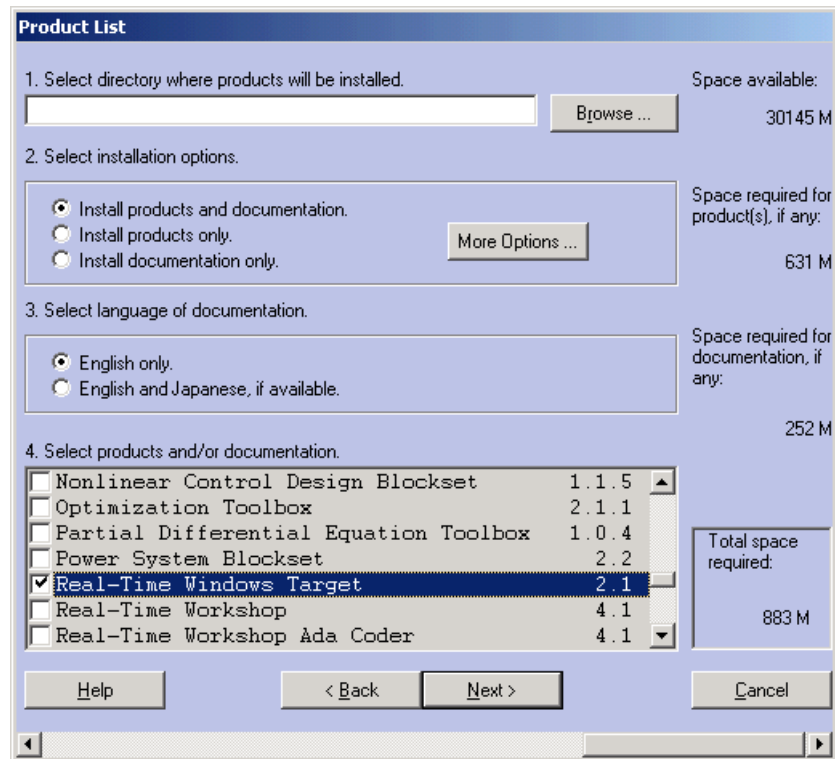
We distribute the Real-Time Windows Target (Version 2.2) on the MathWorks Release 13 CD with the general installation program.

After you get a valid Personal License Password (PLP), you can install the Real-Time Windows Target software. For detailed information about the installation process, see the MATLAB Installation Guide for Windows documentation:

- 1 Insert the Release 13 CD into your host CD-ROM drive.

The installation program starts automatically after a few seconds. If the installation program does not start automatically, run `setup.exe` on the CD.

During the installation process a screen similar to the one shown below allows you to select which products to install.



2 Follow the instructions on each dialog box.

The Real-Time Windows Target installation is now complete.

Your next task is to install the Real-Time Windows Target kernel. See “Installing the Kernel” on page 2-14.

Installing from the Web

We distribute the Real-Time Windows Target Version 2.2 as a single, self-extracting file. This file is not an update, but a complete product, and does not require you to install the Real-Time Windows Target Version 1.0 or 1.5.

Also, this file includes the Real-Time Windows Target documentation as a PDF file.

After you get a valid Personal License Password (PLP), you can install the Real-Time Windows Target on your computer:

- 1 In the Web browser window, enter the following address:

`http://www.mathworks.com`

- 2 On the right side of the page, click the link labeled **Downloads**. On the Downloads Web page, click the link labeled **download products**.

The MATLAB Access Web page opens.

- 3 Enter your last name and your MATLAB Access number. Click the **Login** button.

The Downloads Web page opens.

- 4 From the left list, select the **Windows** check box, and then click the **Continue** button. From the **Select Your Products** list, select the **Real-Time Windows Target 2.2** check box, and then click the **Continue** button.

- 5 On the next Web page, click the **Real-Time Windows Target** link. In the **File Download** dialog box, select **Save this file to disk**, and select the directory where you installed MATLAB.

Your browser downloads the file `Real-Time_Windows_Target.exe` to your computer.

- 6 Double-click the self-extracting file `Real-Time_Windows_Target.exe`.

The install program copies extracted files to a temporary directory and starts the MATLAB installation program.

- 7 Follow the instructions on each dialog box.

After MATLAB finishes the installation, the install program deletes all of the files from the temporary directory.

The Real-Time Windows Target installation is now complete.

Note A PDF version of the Real-Time Windows Target documentation is located at `MATLABROOT\help\pdf_doc\rtwin\rtwin_target_ug.pdf`. You need Adobe Acrobat Reader to view and print this document. You can download Acrobat Reader from the Web at www.adobe.com.

Your next task is to install the Real-Time Windows Target kernel. See “Real-Time Windows Target Kernel” on page 2-14.

Files on the Your Computer

When using the Real-Time Windows Target, you may find it helpful to know where files are located:

- **MATLAB working directory** — Simulink models (`model.mdl`) and the Real-Time Windows Target executable (`model.rwd`)

Note Select a working directory outside of the MATLAB root. See “Initial Working Directory” on page 2-11.

- **Real-Time Workshop project directory** — The Real-Time Workshop C-code files (`model.c`, `model.h`) are in a subdirectory called `model_rtwin`.

Real-Time Windows Target Files — The files included with the Real-Time Windows Target are located in the directory

```
matlabroot\toolbox\rtw\targets\rtwin
```

The Real-Time Windows Target provides files to help Real-Time Workshop generate C code from your Simulink model, and compile that code to a real-time executable:

- **System Target File** (`rtwin.tlc`) — Defines the process of generating C code for the Real-Time Windows Target.
- **Template Makefile and Makefile** (`rtwintmf.m`, `model_name.mk`) — The template makefile serves as a template for generating the real makefile, which the make utility uses during model compilation. During the automatic build procedure, the `make` command extracts information from the template makefile `rtwintmf.m` and generates the makefile `model_name.mk`.
- **Make Command** (`make_rtw.m`) — The standard `make` command supplied with Real-Time Workshop.

Other files provided with the Real-Time Windows Target include

- **I/O drivers** (`*rwd`) — Binaries for I/O device drivers. The Real-Time Windows Target does not link the driver object files with your real-time executable. The drivers are loaded into memory and run by the kernel separately.
- **Simulink external mode interface** (`rtwinext.dll`) — MEX-file for communicating between Simulink external mode and the Real-Time Windows Target kernel.

Simulink external mode uses the MEX-file interface module to download new parameter values to the real-time model and to retrieve signals from the real-time model. You can display these signals in Simulink Scope blocks.

- **Kernel install and uninstall commands** (`rtwintgt.m`, `rtwho.m`) — M-file scripts to install and uninstall the Real-Time Windows Target kernel and check installation.

Initial Working Directory

You should set your MATLAB working directory outside of the MATLAB root directory. The default MATLAB root directory is `c:\matlab`.

This section includes the following topics:

- “Setting the Working Directory from Desktop Icon” on page 2-11
- “Setting the Working Directory from MATLAB” on page 2-11

Setting the Working Directory from Desktop Icon

Your initial working directory is specified in the shortcut file you use to start MATLAB. To change this initial directory, use the following procedure:

- 1 Right-click the MATLAB desktop icon, or from the program menu, right-click the MATLAB shortcut.
- 2 Click **Properties**. In the **Start in** text box, and enter the directory path you want MATLAB to initially use outside of the MATLAB root directory.
- 3 Click **OK**, and then start MATLAB. To check your working directory, type `pwd` or `cd`

Setting the Working Directory from MATLAB

Use the following procedure as an alternative, but temporary, procedure for setting your MATLAB working directory:

- 1 In the MATLAB command window, type `cd c:\mwd`
- 2 Check the current working directory, type `cd`

MATLAB displays

```
ans = c:\mwd or c:\mwd
```

Third Party C Compiler

The Real-Time Windows Target requires one of the following C compilers not included with the Real-Time Windows Target:

- **Microsoft Visual C/C++ compiler** — Version 5.0, 6.0, or 7.0.
- **Watcom C/C++ compiler** — Version 10.6 and 11.0. During installation of your Watcom C/C++ compiler, be sure to specify a DOS target in addition to a Windows target to have the necessary libraries available for linking.

After installation, run the MEX utility to select your compiler as the default compiler for building real-time applications.

Selecting a Default C Compiler

Real-Time Workshop uses the default C compiler to generate executable code, and the MEX utility uses this compiler to create MEX-files.

Use this procedure to select either a Microsoft Visual C/C++ compiler or a Watcom C/C++ compiler before you build an application. Note, the LCC compiler is not supported:

- 1 In the MATLAB window, type

```
mex -setup
```

MATLAB displays the following message:

```
Please choose your compiler for building external interface
(MEX) files. Would you like mex to locate installed
compilers?([y]/n):
```

- 2 Type

```
y
```

MATLAB displays the following message:

```
Select a compiler:
[1] : WATCOM compiler in c:\watcom
[2] : Microsoft compiler in c:\visual
[0] : None
```

```
Compiler:
```

3 Type a number. For example, to select the Microsoft compiler, type

2

MATLAB displays the message:

Please verify your choices:

Compiler: Microsoft 5.0

Location: c:\visual

Are these correct?([y]/n)

4 Type

y

MATLAB resets the default compiler and displays the message:

The default options file:

C:\WINNT\Profiles\username\Application

Data\MathWorks\MATLAB\mexopts.bat is being updated.

Real-Time Windows Target Kernel

A key component of the Real-Time Windows Target is a real-time kernel that interfaces with the Windows operating system in a way that allows your real-time executable to run at your selected sample rate. The kernel assigns the highest priority of execution to your real-time executable.

This section includes the following topics:

- “Installing the Kernel” on page 2-14
- “Uninstalling the Kernel” on page 2-16

Installing the Kernel

During installation, all software for the Real-Time Windows Target is copied onto your hard drive. The kernel, although copied to the hard drive, is not automatically installed. Installing the kernel sets up the kernel to start running in the background each time you start your computer.

After you install the Real-Time Windows Target (version 2.2), you can install the kernel. You need to install the kernel before you can run a Real-Time Windows Target executable:

- 1 In the MATLAB command window, type

```
rtwintgt -install
```

MATLAB displays the message

```
You are going to install the Real-Time Windows Target kernel.  
Do you want to proceed? [y] :
```

- 2 Continue installing the kernel. Type

```
y
```

MATLAB installs the kernel and displays the message

```
The Real-Time Windows Target kernel has been successfully  
installed.
```

If a message is displayed asking you to restart your computer, you need to restart your computer before the kernel runs correctly.

3 Check that the kernel was correctly installed. Type

```
rtwho
```

MATLAB should display a message similar to

```
Real-Time Windows Target version 2.2 (C) The MathWorks, Inc.  
1994-2002  
MATLAB performance = 100.0%  
Kernel timeslice period = 1 ms
```

Once the kernel is installed, you can leave it installed. After you have installed the kernel, it remains idle, which allows Windows to control the execution of any standard Windows application. Standard Windows applications include internet browsers, word processors, and MATLAB.

It is only during real-time execution of your model that the kernel intervenes to ensure that your model is given priority to use the CPU to execute each model update at the prescribed sample intervals. Once the model update at a particular sample interval completes, the kernel releases the CPU to run any other Windows application that may need servicing.

Uninstalling the Kernel

If you encounter any problems with the Real-Time Windows Target, you can uninstall the kernel. The kernel executable file remains on your hard drive so that you can reinstall it:

- 1 In the MATLAB command window, type

```
rtwintgt -uninstall
```

MATLAB displays the message

```
You are going to uninstall the Real-Time Windows Target kernel.  
Do you want to proceed? [y] :
```

- 2 To continue uninstalling the kernel, type

```
y
```

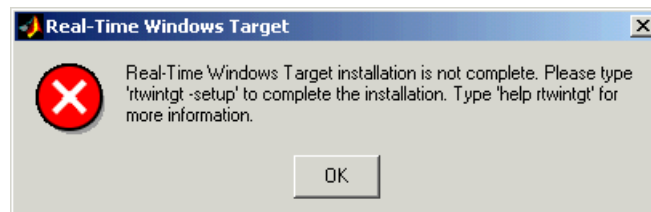
MATLAB uninstalls the kernel by removing it from memory and displays the message.

```
The Real-Time Windows Target kernel has been successfully  
uninstalled.
```

- 3 To check that the kernel was correctly uninstalled, type

```
rtwho
```

MATLAB should display the following message



If you are running Windows 98, you need to reboot your computer before the uninstall is complete.

Once uninstalled, the kernel is no longer active, and has no impact on the operation of your computer.

There are two additional ways to uninstall the Real-Time Windows Target kernel. They are useful if you uninstall MATLAB before you uninstall the kernel.

From the **Start** menu select **Settings, Control Panel**, and **Add/Remove Software**. Choose **Real-Time Windows Target** to uninstall the kernel.

Alternately, from the DOS prompt of your computer, type

```
rtwintgt -uninstall
```

and the kernel will uninstall from your system. Typing

```
rtwintgt -forceuninstall
```

forcibly deregisters the kernel from the operating system without deleting any files. This option should only be used when all other attempts to uninstall the kernel fail. This command can be used both within MATLAB and at the DOS prompt.

Testing the Installation

The Real-Time Windows Target includes several demo models. You can use one of the demo models to test your installation. Demo models simplify testing of your installation since they are configured with settings that include the correct target, scope settings, sample time, and integration algorithm.

Once you have completed the installation of the Real-Time Windows Target and the kernel, we recommend a quick test by at least running the model `rtvdp.mdl`. If you change your installation or compiler, we also recommend doing this test as a quick check to confirm that the Real-Time Windows Target is still working.

This section includes the following topics:

- “Running the Model `rtvdp.mdl`” on page 2-18
- “Displaying Status Information” on page 2-21
- “Detecting Excessive Sample Rates” on page 2-22
- “Demo Library” on page 2-22

Running the Model `rtvdp.mdl`

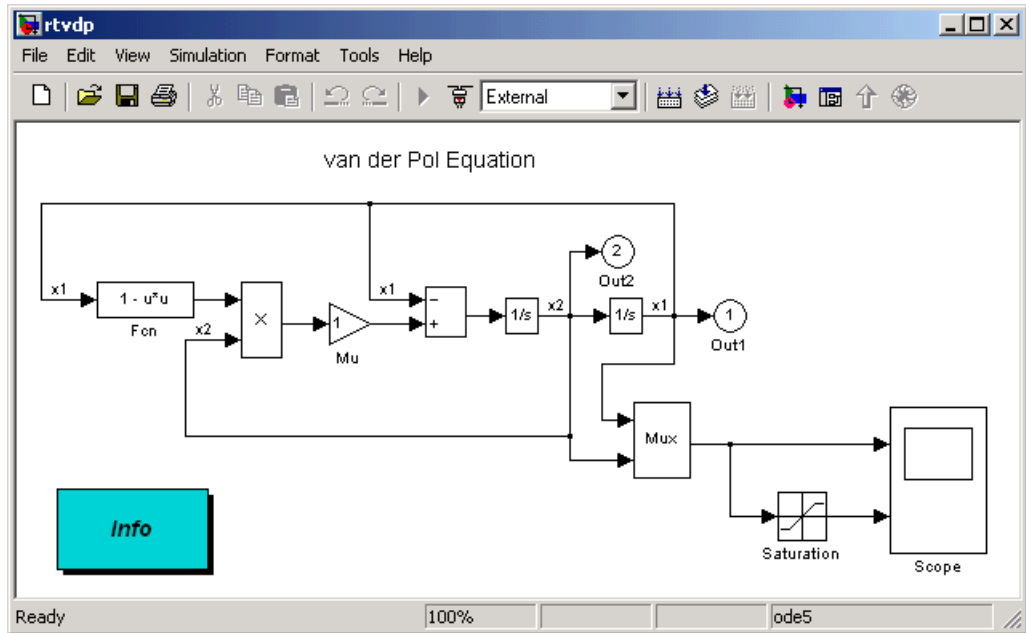
The model `rtvdp.mdl` does not have any I/O blocks so that you can run this model regardless of the I/O boards in your computer. Running this model will test the installation by running Real-Time Workshop, your third-party C compiler, the Real-Time Windows Target, and the Real-Time Windows Target kernel.

After you have installed the Real-Time Windows Target kernel, you can test the entire installation by building and running a real-time application. The Real-Time Windows Target includes the model `rtvdp.mdl`, which already has the correct Real-Time Workshop options selected for you:

1 In the MATLAB command window, type

```
rtvdp
```

The Simulink model `rtvdp.mdl` window opens.



- 2** From the **Tools** menu, point to **Real-Time Workshop**, and then click **Build Model**.

The MATLAB command window displays the following messages:

```

### Starting Real-Time Workshop build for model: rtdvp
### Invoking Target Language Compiler on rtdvp.rtw
. . .
### Compiling rtdvp.c
. . .
### Created Real-Time Windows Target module rtdvp.rwd.
### Successful completion of Real-Time Workshop build procedure
for model: rtdvp

```

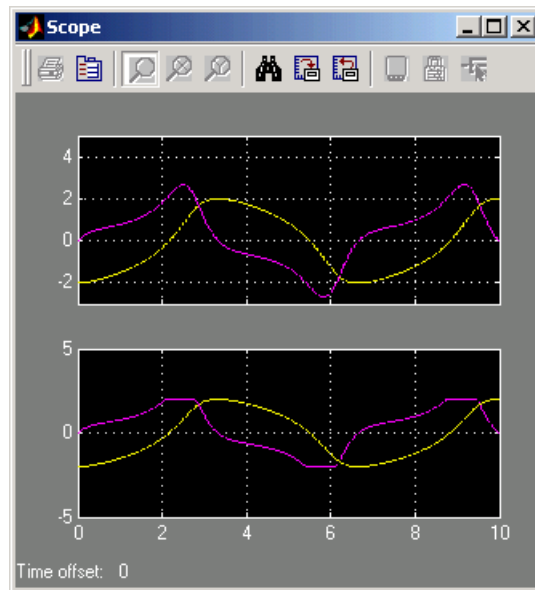
- 3 From the **Simulation** menu, click **External**, and then click **Connect to target**.

The MATLAB command window displays the following message:

```
Model rtvdp loaded
```

- 4 From **Simulation** menu, click **Start real-time code**.

The Scope window displays the output signals. If your Scope window looks like the figure shown below, then you have successfully installed the Real-Time Windows Target and have run a real-time application.



- 5 From **Simulation** menu, click **Stop real-time code**.

The real-time application stops running, and the Scope window stops displaying the output signals.

Displaying Status Information

The Real-Time Windows Target provides the command `rtwho` for accessing the kernel and displaying status information. It lists information about the version number, kernel performance, and history variables. Also, you can determine whether or not the Real-Time Windows kernel is presently installed:

- 1 In the MATLAB command window, type

```
rtwho
```

MATLAB displays messages similar to those shown below.

```
Real-Time Windows Target version 2.2 (C) The MathWorks, Inc.
1994-2002
MATLAB performance = 100.0%
Kernel timeslice period = 1 ms
```

```
DRIVERS:           Name      Address  Parameters
                Humusoft AD512      0x300    [ ]
```

- 2 Interpret the message.

This message indicates that MATLAB and other nonreal-time applications (for example, a word processor) are able to run at 100% performance because no real-time applications are currently executing on your PC.

When a real-time application is executing, the MATLAB performance is at a value below 100%. For example, if the MATLAB performance = 90.0%, then the real-time application is using 10% of the CPU time.

We recommend that you select a sample rate so that `rtwho` returns a MATLAB performance of at least 80%.

The kernel time slice period is the current frequency of the hardware timer interrupt. One millisecond is the maximum value for models with large sample times (slow sampling rate) or when an application has not been built. This value changes when you select sampling times less than 1 millisecond.

Detecting Excessive Sample Rates

If your specified sample rate is too fast, the Real-Time Windows Target detects and reports this during real-time execution. Sampling rates exceeding 10 kHz can be achieved on Pentium computers. Once the model is running, the `rtwho` command can be issued in the MATLAB command line to observe the system performance. As indicated, MATLAB performance decreases as the system becomes overloaded:

```
Real-Time Windows Target version 2.2 (C) The MathWorks, Inc.  
1999-2002
```

```
MATLAB performance = 99.1%
```

```
Kernel timeslice period = 0.0999 ms
```

```
TIMERS:  Number   Period   Running
```

```
          1       0.01     Yes
```

```
DRIVERS:           Name   Address  Parameters
```

```
Humusoft AD512     0x300   []
```

```
          ecg       0       []
```

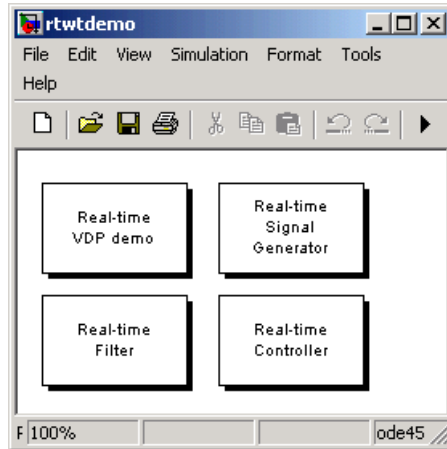
Demo Library

The demo library includes models with preset values and dialog boxes. These models include a configuration of examples that use no I/O, A/D only, A/D and D/A in a simple signal processing demo, as well as in a simple control demo.

Examples that use I/O blocks require you to configure the Adapter block to match the I/O board installed in your computer.

- 1 In the MATLAB command window, type
`rtwtdemo`

The **rtwtdemo** window opens and displays the demo models provided with the Real-Time Windows Target.



- 2 Double-click a demo block to open the model.

Basic Procedures

Simulink Model	3-3
Creating a Simulink Model	3-3
Entering Simulation Parameters for Simulink	3-7
Entering Scope Parameters for Signal Tracing	3-9
Running a Nonreal-Time Simulation	3-12
Real-Time Application	3-14
Entering Simulation Parameters for Real-Time Workshop	3-14
Entering Scope Parameters for Signal Tracing	3-17
Creating a Real-Time Application	3-20
Entering Additional Scope Parameters for Signal Tracing	3-21
Running a Real-Time Application	3-25
Running a Real-Time Application from the MATLAB Command Line	3-28
Signal Logging to the MATLAB Workspace	3-29
Entering Scope Parameters	3-29
Entering Signal and Triggering Properties	3-31
Plotting Logged Signal Data	3-35
Signal Logging to a Disk Drive	3-37
Entering Scope Parameters	3-37
Entering Signal and Triggering Properties	3-40
Entering Data Archiving Parameters	3-43
Plotting Logged Signal Data	3-46
Parameter Tuning	3-48
Types of Parameters	3-48
Changing Model Parameters	3-49

The basic procedures explain how to create a Simulink or real-time application, and how to run a simulation or execution.

This chapter includes the following sections:

- **Simulink Model** — Create a Simulink model and run a nonreal-time simulation
- **Real-Time Application** — Create a real-time application, generate code from that model, and run a real-time execution
- **Signal Logging to the MATLAB Workspace** — Save data from a simulation or execution, and then analyze or visualize that data
- **Signal Logging to a Disk Drive** — Save data from a real-time execution, and then analyze or visualize that data
- **Parameter Tuning** — Change parameters in your application while it is running in real time

Simulink Model

A Simulink model is a graphical representation of your physical system. You create a Simulink model for nonreal-time simulation of your system, and then you use the Simulink model to create a real-time application.

This section includes the following topics:

- “Creating a Simulink Model” on page 3-3
- “Entering Simulation Parameters for Simulink” on page 3-7
- “Entering Scope Parameters for Signal Tracing” on page 3-9
- “Running a Nonreal-Time Simulation” on page 3-12

Creating a Simulink Model

This procedure explains how to create a simple Simulink model. You use this model as an example to learn other procedures in the Real-Time Windows Target.

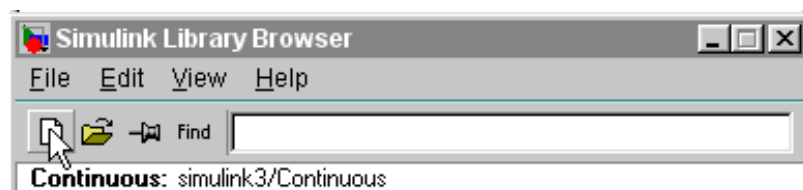
You need to create a Simulink model before you can run a simulation, or create a real-time application:

- 1 In the MATLAB command window, type

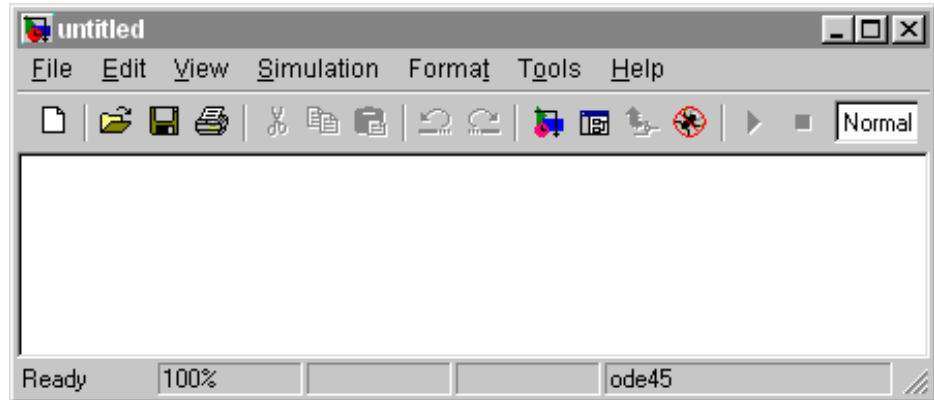
```
simulink
```

The Simulink Library Browser window opens.

- 2 From the toolbar, click the **Create a new model** button.



An empty Simulink window opens.



- 3** In the Simulink Library Browser window, double-click **Simulink**, and then double-click **Sources**. Click-and-drag **Signal Generator** to the Simulink window.

Double-click **Continuous**. Click-and-drag **Transfer Fcn** to the Simulink window.

Double-click **Sinks**. Click-and-drag **Scope** to the Simulink window.

- 4** Connect the **Signal Generator** output to the **Transfer Fcn** input by clicking-and-dragging a line between the blocks. Likewise, connect the **Transfer Fcn** output to the **Scope** input.

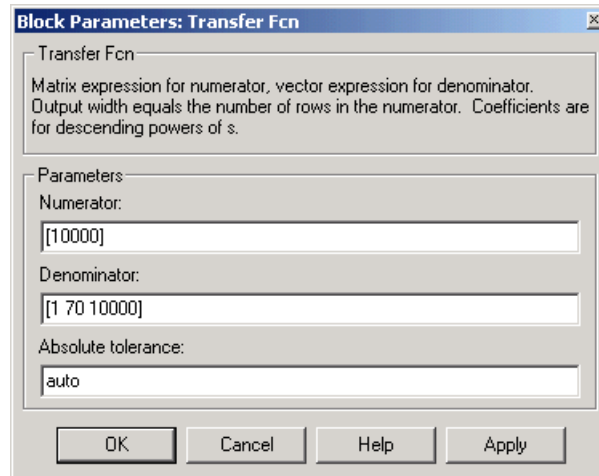
- 5** Double-click the **Transfer Fcn** block. The **Block Parameters** dialog box opens. In the **Numerator** text box, enter

[10000]

In the **Denominator** text box, enter

[1 70 10000]

Your **Block Parameters** dialog box will look similar to the figure shown below.



- 6 Click **OK**.
- 7 Double-click the Signal Generator block. The **Block Parameters** dialog box opens. From the **Wave form** list, select square.

In the **Amplitude** text box, enter

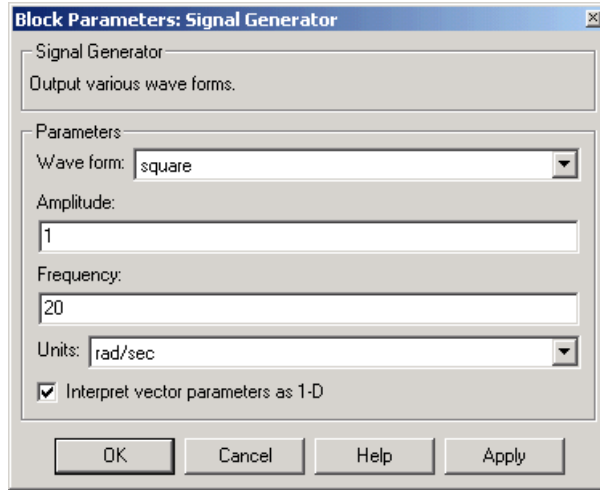
1

In the **Frequency** text box, enter

20

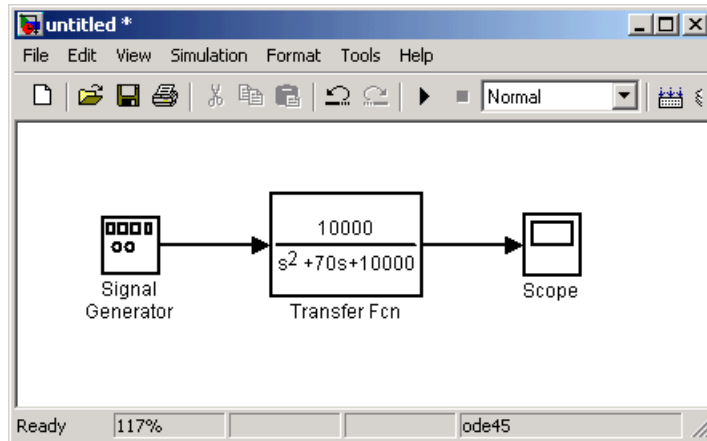
From the **Units** list, select rad/sec.

Your **Block Parameters** dialog box will look similar to the figure shown below.



8 Click **OK**.

The completed Simulink block diagram is shown below.



- 9 From the **File** menu, click **Save As**. The **Save As** dialog box opens. In the **File name** text box, enter a filename for your Simulink model and click **Save**. For example, type

```
rtwin_model
```

Simulink saves your model in the file `rtwin_model.mdl`.

Entering Simulation Parameters for Simulink

The simulation parameters give information to Simulink for running a simulation.

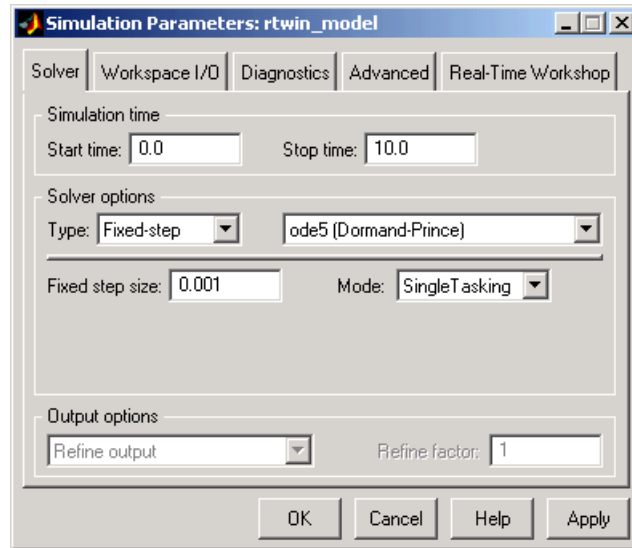
After you create a Simulink model, you can enter the simulation parameters for Simulink. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model:

- 1 In the Simulink window, and from the **Simulation** menu, click **Simulation parameters**. In the **Simulation Parameters** dialog box, click the **Solver** tab.

The Solver pane opens.

- 2 In the **Start time** box, enter 0.0. In the **Stop time** box, enter the amount of time you want your model to run. For example, enter 10.0 seconds.
- 3 From the **Type** list, choose **Fixed-step**. Real-Time Workshop does not support variable step solvers.
- 4 From the integration algorithm list, choose a solver. For example, choose the general purpose solver `ode5 (Dormand-Prince)`.
- 5 In the **Fixed step size** box, enter a sample time. For example, enter 0.001 seconds for a sample rate of 1000 samples/second.
- 6 From the **Mode** list, choose `SingleTasking`. For models with blocks that have different sample times, choose `MultiTasking`.

Your **Solver** pane will look similar to the figure shown below.



7 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the **Simulation Parameters** dialog box.

Entering Scope Parameters for Signal Tracing

You enter or change scope parameters to specify the x -axis and y -axis in a Scope window. Other properties include the number of graphs in one Scope window and the sample time for models with discrete blocks.

After you add a Scope block to your Simulink model, you can enter the scope parameters for signal tracing:

- 1 In the Simulink window, double-click the Scope block.

A Scope window opens.

- 2 Click the **Parameters** button.



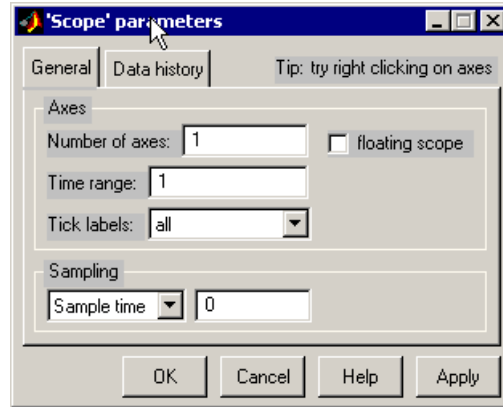
A **Scope parameters** dialog box opens.

- 3 Click the **General** tab. In the **Number of axes** box, enter the number of graphs you want in one Scope window. For example, enter 1 for a single graph. Do not select the **floating scope** check box.

In the **Time range** box, enter the upper value for the time range. For example, enter 1 second. From the **Tick labels** list, choose all.

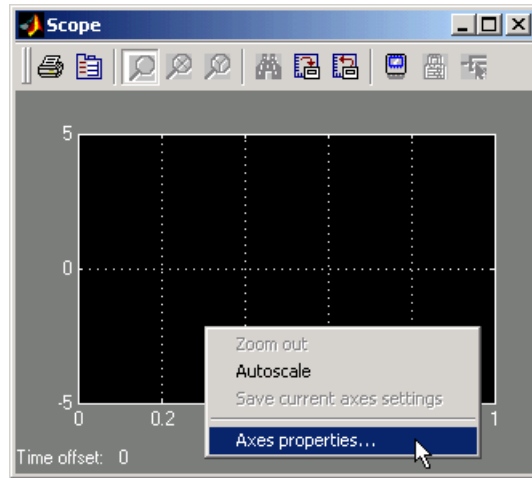
From the **Sampling** list, choose `Sample time` and enter 0 in the text box. Entering 0 indicates that Simulink evaluates this block as a continuous time block. If you have discrete blocks in your model, enter the **Fixed step size** you entered in the **Simulation Parameters** dialog box.

Your **Scope parameters** dialog box will look similar to the figure shown below.

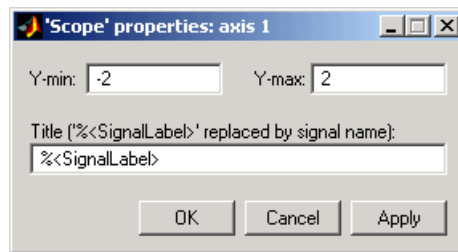


- 4 Do one of the following:
- Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the **Scope parameters** dialog box.

- 5 In the Scope window, point to the y-axis shown in the figure below, and right-click.



- 6 From the pop-up menu, click **Axis Properties**.
- 7 The **Scope properties: axis 1** dialog box opens. In the **Y-min** and **Y-max** text boxes, enter the range for the y-axis in the Scope window. For example, enter -2 and 2 as shown in the figure below.



- 8 Do one of the following:
 - Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the **Simulation Parameters** dialog box.

Running a Nonreal-Time Simulation

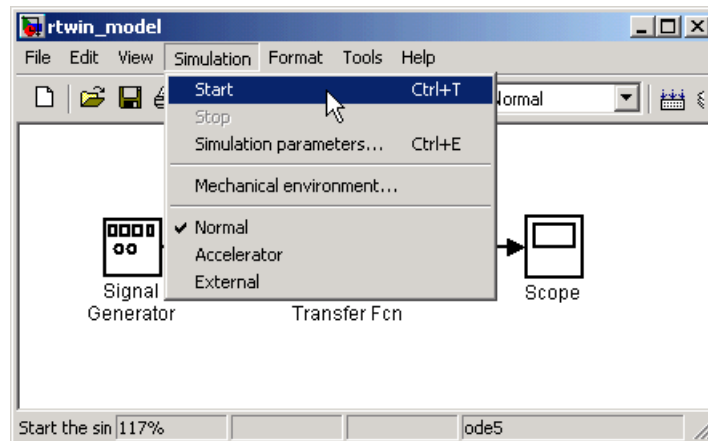
You use Simulink in normal mode to run a nonreal-time simulation. Running a simulation lets you observe the behavior of your model in nonreal-time.

After you load your Simulink model into the MATLAB workspace, you can run a simulation. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have loaded that model:

- 1 In the Simulink window, double-click the Scope block.

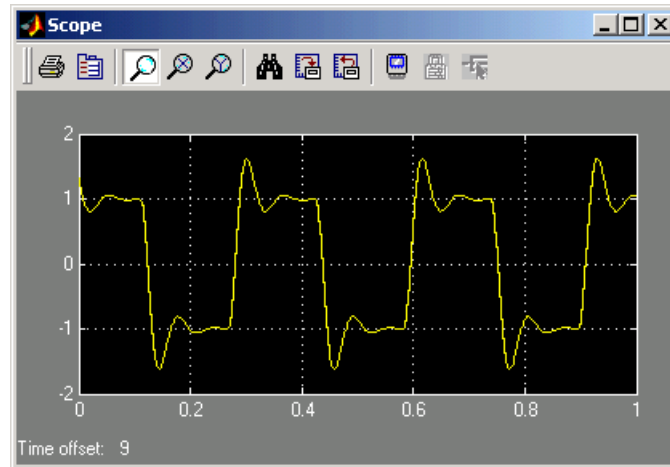
Simulink opens a Scope window with an empty graph.

- 2 From the **Simulation** menu, click **Normal**, and then click **Start**.



Simulink runs the simulation and plots the signal data in the Scope window.

During the simulation, the Scope window displays the samples for one time range, increases the time offset, and then displays the samples for the next time range.



- 3 Do one of the following:
 - Let the simulation run to the stop time.
 - From the **Simulation** menu, click **Stop**.

The simulation stops. MATLAB does not display any messages.

Real-Time Application

You create a real-time application to let your system run while synchronized to a real-time clock. This allows your system to control or interact with an external system. This is necessary if you use your system to stabilize a physical plant.

The process of creating and running a real-time application includes the creation of a Simulink Model from the previous section:

- “Creating a Simulink Model” on page 3-3
- “Entering Simulation Parameters for Simulink” on page 3-7

This section includes the following topics:

- “Entering Simulation Parameters for Real-Time Workshop” on page 3-14
- “Entering Scope Parameters for Signal Tracing” on page 3-17
- “Creating a Real-Time Application” on page 3-20
- “Entering Additional Scope Parameters for Signal Tracing” on page 3-21
- “Running a Real-Time Application” on page 3-25
- “Running a Real-Time Application from the MATLAB Command Line” on page 3-28

Entering Simulation Parameters for Real-Time Workshop

The simulation parameters are used by Real-Time Workshop for generating C code and building a real-time application.

After you create a Simulink model, you can enter the simulation parameters for Real-Time Workshop. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model:

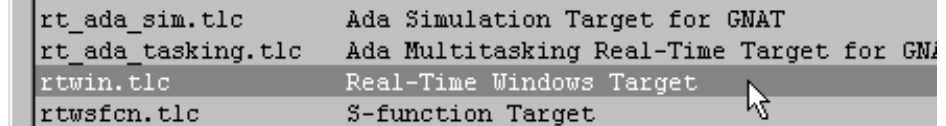
- 1 In the Simulink window, and from the **Simulation** menu, click **Simulation parameters**. In the **Simulation Parameters** dialog box, click the **Real-Time Workshop** tab.

The Real-Time Workshop pane opens.

- 2 Click the **Browse** button.

The System Target File Browser opens.

- 3 Select the system target file for the Real-Time Windows Target.



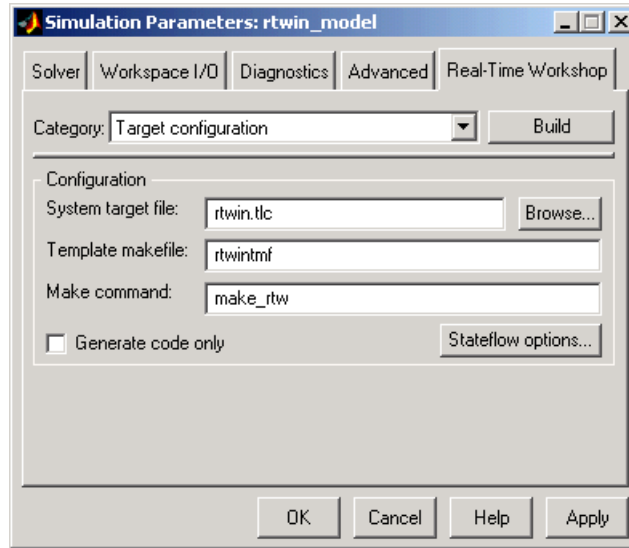
rt_ada_sim.tlc	Ada Simulation Target for GNAT
rt_ada_tasking.tlc	Ada Multitasking Real-Time Target for GNAT
rtwin.tlc	Real-Time Windows Target
rtwsfcn.tlc	S-function Target

- 4 Click **OK**.

The system target file `rtwin.tlc`, the template makefile `rtwintmf`, and the make command `make_rtw` are automatically entered into the Real-Time Workshop pane.

Although not visible in the Real-Time Workshop pane, the external target interface MEX file `rtwinext` is also configured after you click **OK**. This allows external mode to pass new parameters to the real-time application and to return signal data from the real-time application. The data is displayed in Scope blocks or saved with signal logging.

Your Real-Time Workshop pane will look similar to the figure shown below.



Do not select the **Inline parameters** check box on the **Advanced** tab. Inlining parameters is used for custom targets when you want to reduce the amount of RAM or ROM with embedded systems. Also, if you select inlining parameters, the parameter tuning feature is disabled. Since PCs have more memory than embedded systems, we recommend that you do not inline parameters.

- 5 Do one of the following:
 - Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the **Simulation Parameters** dialog box.

Entering Scope Parameters for Signal Tracing

You enter or change scope parameters to format the x-axis and y-axis in a Scope window. Other parameters include the number of graphs in a one Scope window and whether the scope is connected to a continuous or discrete model.

If you entered the scope parameters for running a simulation, you can skip this procedure. This information is repeated here if you did not run a simulation.

After you add a Scope block to your Simulink model, you can enter the scope parameters for signal tracing:

- 1 In the Simulink window, double-click the Scope block.

A Scope window opens.

- 2 Click the **Parameters** button.



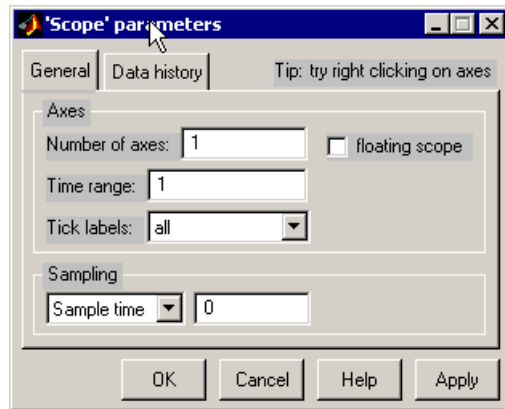
A **Scope parameters** dialog box opens.

- 3 Click the **General** tab. In the **Number of axes** box, enter the number of graphs you want in one Scope window. For example, enter 1 for a single graph. Do not select the **floating scope** check box.

In the **Time range** box, enter the upper value for the time range. For example, enter 1 second. From the **Tick labels** list, choose all.

From the **Sampling** list, choose **Sample time** and enter 0 in the text box. Entering 0 indicates that Simulink evaluates this block as a continuous time block. If you have discrete blocks in your model, enter the **Fixed step size** you entered in the **Simulation Parameters** dialog box.

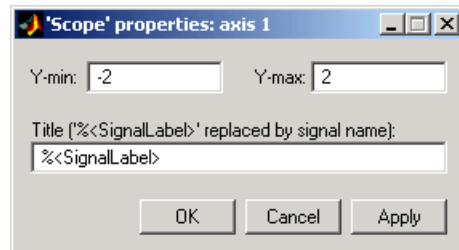
Your **Scope parameters** dialog box will look similar to the figure shown below.



- 4 Do one of the following:
 - Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the **Scope parameters** dialog box.
- 5 In the Scope window, point to the y-axis and right-click. From the menu, click **Axis Properties**.

The **Scope properties: axis 1** dialog box opens.

- 6 In the **Y-min** and **Y-max** text boxes enter the range for the *y*-axis in the Scope window. For example, enter -2 and 2.



- 7 Do one of the following:
- Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the **Scope properties: axis 1** dialog box.

Creating a Real-Time Application

Real-Time Workshop generates C code from your Simulink model, then a third-party C compiler compiles and links that C code into a real-time application.

After you enter parameters into the **Simulation Parameters** dialog box for Real-Time Workshop, you can build a real-time application. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you have loaded that model:

- 1** In the Simulink window, and from the **Tools** menu, point to **Real-Time Workshop**, and then click **Build Model**.

The build process does the following:

- Real-Time Workshop creates the C code source files `rtwin_model.c` and `rtwin_model.h`.
 - The make utility `make_rtw.exe` creates the makefile `rtwin_model.mk` from the template makefile `rtwintmf`.
 - The make utility `make_rtw.exe` builds the real-time application `rtwin_model.rwd` using the makefile `rtwin_model.mk` created above. The file `rtwin_model.rwd` is a binary file that we refer to as your real-time application. You can run the real-time application with the Real-Time Windows Target kernel.
- 2** Connect your Simulink model to your real-time application. See “Entering Additional Scope Parameters for Signal Tracing” on page 3-21.

After you create a real-time application, you can exit MATLAB, start MATLAB again, and then connect and run the executable without having to rebuild.

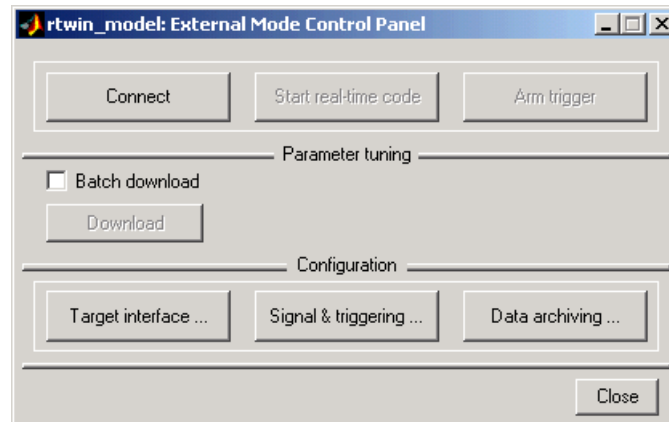
Entering Additional Scope Parameters for Signal Tracing

Simulink external mode connects your Simulink model to your real-time application. This connection allows you to use the Simulink block diagram as a graphical user interface to your real-time application.

After you have created a real-time application, you can enter scope parameters for signal tracing with Simulink external mode:

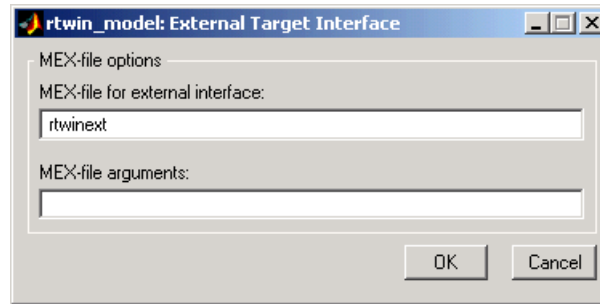
- 1 In the Simulation window, and from the **Tools** menu, click **External Mode Control Panel**.

The **External Mode Control Panel** dialog box opens.



- 2 Click the **Target Interface** button.

The **External Target Interface** dialog box opens.



- 3** In the **MEX-file for external interface** box, enter `rtwinext`.

The MEX-file, `rtwinext.dll`, is supplied with the Real-Time Windows Target to work with Simulink external mode and support uploading signal data and downloading parameter values.

- 4** Click **OK**.
- 5** Click the **Signal & Triggering** button.

The **External Signal & Triggering** dialog box opens.

- 6** Click the **Select all** button. From the **Source** list, choose `manual`. From the **Mode** list, choose `normal`.

The X under **Signal selection** designates that a signal has been tagged for data collection, and T designates that the signal has been tagged as a trigger signal.

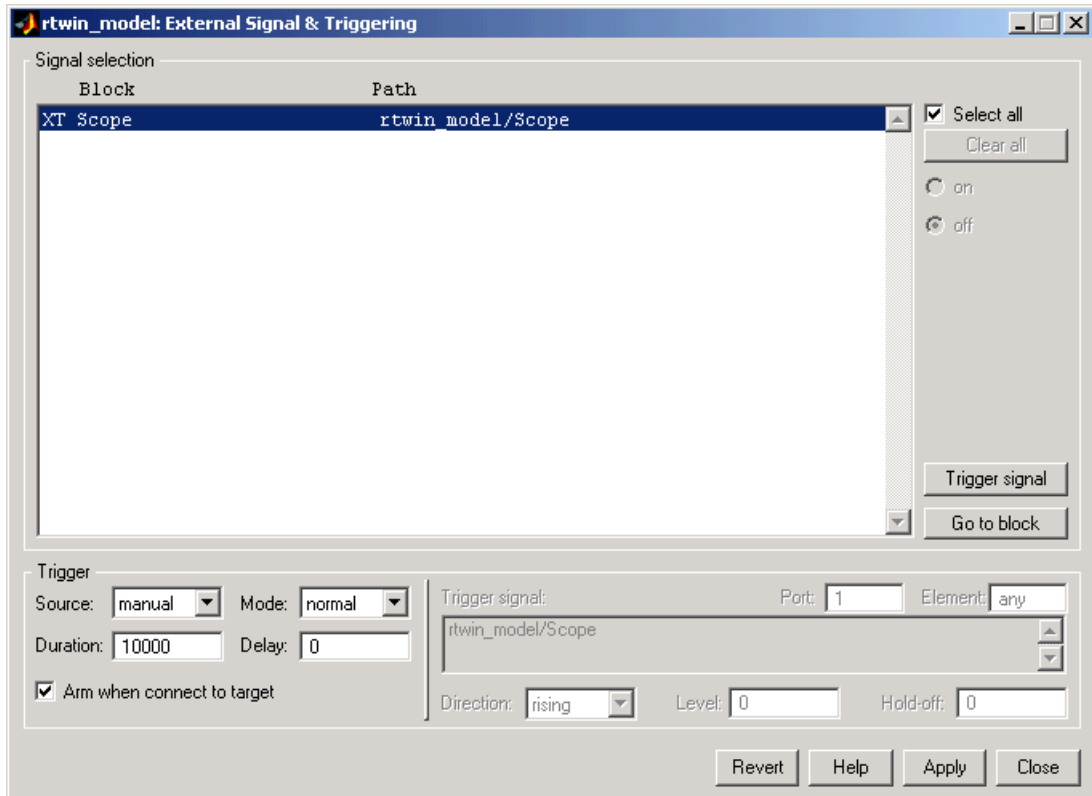
- 7** In the **Duration** box, enter the number of sample points in a data buffer. For example, if you have a sample rate of 1000 samples/second and a stop time of 10 seconds, you could enter

10000

- 8 Select the **Arm when connect to target** check box.

Note If you do not select this check box, data is not displayed in the scope window.

The **Signal & Triggering** dialog box will look similar to the figure below.



- 9 Do one of the following:
 - Click **Apply** to apply the changes to your model and leave the dialog box open.

- Click **Close** to apply the changes to your model and close the **Simulation Parameters** dialog box.

Note You must click the **Apply** or **Close** button on the **Signal and Triggering** dialog box for the changes you made to take effect. Generally it is not necessary to rebuild your real-time application.


Running a Real-Time Application

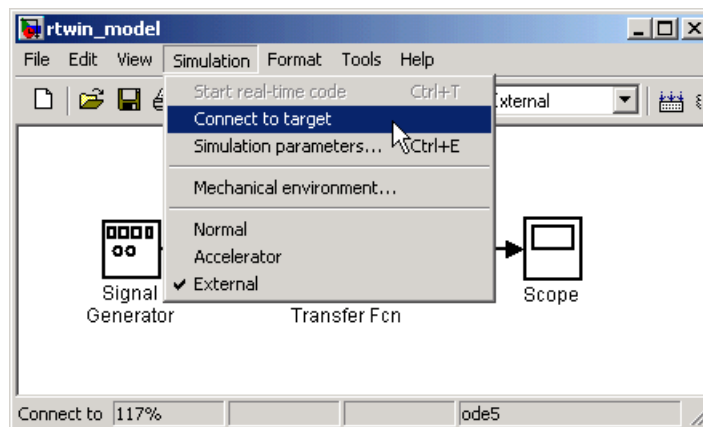
You run your real-time application to observe the behavior of your model in real time with the generated code.

The process of connecting consists of

- Establishing a connection between your Simulink model and the kernel to allow exchange of commands, parameters, and logged data.
- Running the application in real time.


After you build the real-time application, you can run your model in real time. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you have created a real-time application for that model:

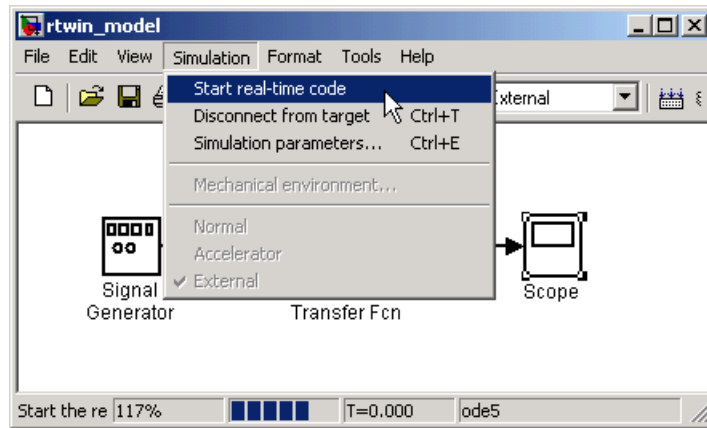
- 1 From the **Simulation** menu, click **External**, and then click **Connect to target**. Also, you can connect to the target from the toolbar by clicking .



MATLAB displays the message

```
Model rtwin_model loaded
```

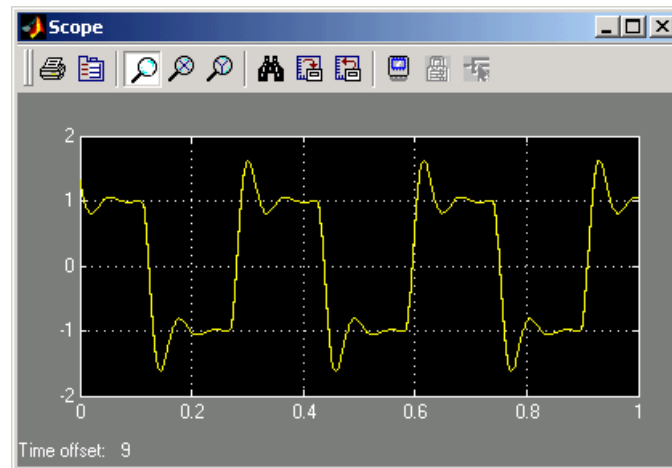
- 2 In the Simulation window, and from the **Simulation** menu, click **Start real-time code**. Also, you can start the execution from the toolbar by clicking .



Simulink runs the execution and plots the signal data in the Scope window.

In this example, the Scope window displays 1000 samples in 1 second, increases the time offset, and then displays the samples for the next 1 second.

Note Transfer of data is less critical than calculating the signal outputs at the selected sample interval. Therefore, data transfer runs at a lower priority in the remaining CPU time after real-time application computations are performed while waiting for another interrupt to trigger the next real-time application update. The result may be a loss of data points displayed in the Scope window.



3 Do one of the following:

- Let the execution run until it reaches the stop time.
- From the **Simulation** menu, click **Stop real-time code**.

The real-time application stops, and MATLAB displays the message

```
Model rtwin_model unloaded
```

Running a Real-Time Application from the MATLAB Command Line

You can use the MATLAB command line interface as an alternative to using the Simulink GUI. Enter commands directly in the MATLAB command line window or enter them in an M-file.

After you build the real-time application, you can run your model in real time. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you have created a real-time application for that model:

- 1 In the MATLAB command line, type

```
set_param(gcs, 'SimulationMode', 'external')
```

Simulink changes to external mode.

- 2 Type,

```
set_param(gcs, 'SimulationCommand', 'connect')
```

MATLAB loads the real-time application, connects it to the Simulink block diagram, and displays the message.

```
Model rtwin_model loaded
```

- 3 Type,

```
set_param(gcs, 'SimulationCommand', 'start')
```

Simulink starts running the real-time application.

- 4 Type,

```
set_param(gcs, 'SimulationCommand', 'stop')
```

Simulink stops the real-time application.

Signal Logging to the MATLAB Workspace

Signal logging is the process of saving (logging) data to a variable in your MATLAB workspace or to a MAT-file on your disk drive. This allows you to use MATLAB functions for data analysis and MATLAB plotting functions for visualization. You can save data to a variable during a simulation or during an execution.

To use signal logging with the Real-Time Windows Target, you must add a Scope block to your Simulink model.

This section includes the following topics:

- “Entering Scope Parameters” on page 3-29
- “Entering Signal and Triggering Properties” on page 3-31
- “Plotting Logged Signal Data” on page 3-35

Simulink external mode does not support data logging with Outport blocks in your Simulink model. This means you do not enter or select parameters on the **Workspace I/O** pane in the **Simulation Parameters** dialog box.

Entering Scope Parameters

Data is saved to the MATLAB workspace through a Simulink Scope block. Scope block parameters need to be set for data to be saved.

After you create a Simulink model and add a Scope block, you can enter the scope parameters for signal logging to the MATLAB workspace. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model.

Note If you entered the scope parameters for running a simulation, you may want to look over this procedure because the **Scope parameters** dialog box is related to the **External Signal and Triggering** dialog box.

1 In the Simulink window, double-click the Scope block.

A Scope window opens.

- 2 On the toolbar, click the **Parameters** button.



A **Scope parameters** dialog box opens.

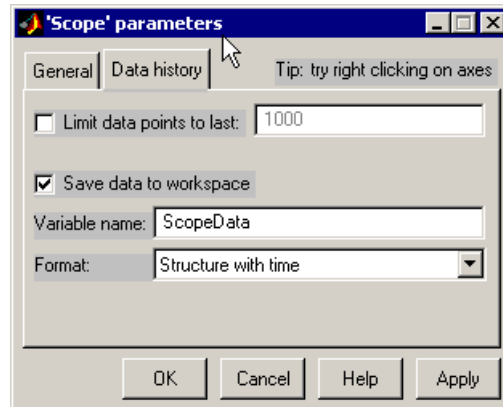
- 3 Click the **Data history** tab.
- 4 Do one of the following:
- If you are running a simulation, you can select the **Limit data points to last** check box, and enter the number of sample points to save.
 - If you are running an execution, do not select the **Limit data points to last** check box.

Note The **Limit data points to last** check box is related to the **Duration** value in the **External Signal and Triggering** dialog box. The smaller of either value limits the number of sample points saved to the MATLAB workspace. When using the Real-Time Windows Target, we recommend that you use the **Duration** value to set the number of sample points you save.

To set the **Duration** value, see “Entering Signal and Triggering Properties” on page 3-31.

- 5 Select the **Save data to workspace** check box. In the **Variable name** text box, enter the name of a MATLAB variable. The default name is ScopeData.
- 6 From the **Format** list, choose either Structure with time, Structure, or Array (compatible with V2.0-2.2). For example, to save the sample times and signal values at those times, choose Structure with time.

Your Data history pane will look similar to the figure shown below.



7 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the **Scope parameters** dialog box.

Note When you modify anything in the **Scope parameters** dialog box, you must click the **Apply** or **OK** button for the changes to take effect, and you must rebuild your real-time application before connecting and starting it. If you do not rebuild, an error dialog box will open. If you do not click **Apply**, your executable will run, but it will use the old settings.

The reason why you need to rebuild is because the model checksum includes settings from the Scope block used for signal logging. If the model checksum does not match the checksum in the generated code, the real-time application cannot run. Always rebuild your real-time application after changing Scope parameters.

Entering Signal and Triggering Properties

Data is saved to the MATLAB workspace through a Simulink Scope block. Signal and triggering properties need to be set only when running a real-time application. If you are running a simulation, you can skip this procedure.

After you create a Simulink model and add a Scope block, you can enter the signal and triggering properties for logging to the MATLAB workspace. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model:

- 1** In the Simulink window, and from the **Tools** menu, click **External Mode Control Panel**.

The **External Mode control Panel** dialog box opens.

- 2** Click the **Signal & Triggering** button.

The **External Signal & Triggering** dialog box opens.

- 3** Click the **Select all** button. From the **Source** list, choose `manual`. From the **Mode** list, choose `normal`.

The X under **Signal selection** designates that a signal has been tagged for data collection, and T designates that the signal has been tagged as a trigger signal.

- 4** In the **Duration** box, enter the number of sample points in a data buffer. For example, if you have a sample rate of 1000 samples/second and a stop time of 10 seconds, you could enter

10000

Note The **Duration** value is related to the **Limit data points to last** value in the **Scope parameters** dialog box. The smaller of either value limits the number of sample points saved to the MATLAB workspace. We recommend that you do not select the **Limit data points to last** check box, and use the **Duration** value to set the number of sample points saved.

To clear the **Limit data points to last** check box, see “Entering Scope Parameters” on page 3-29.

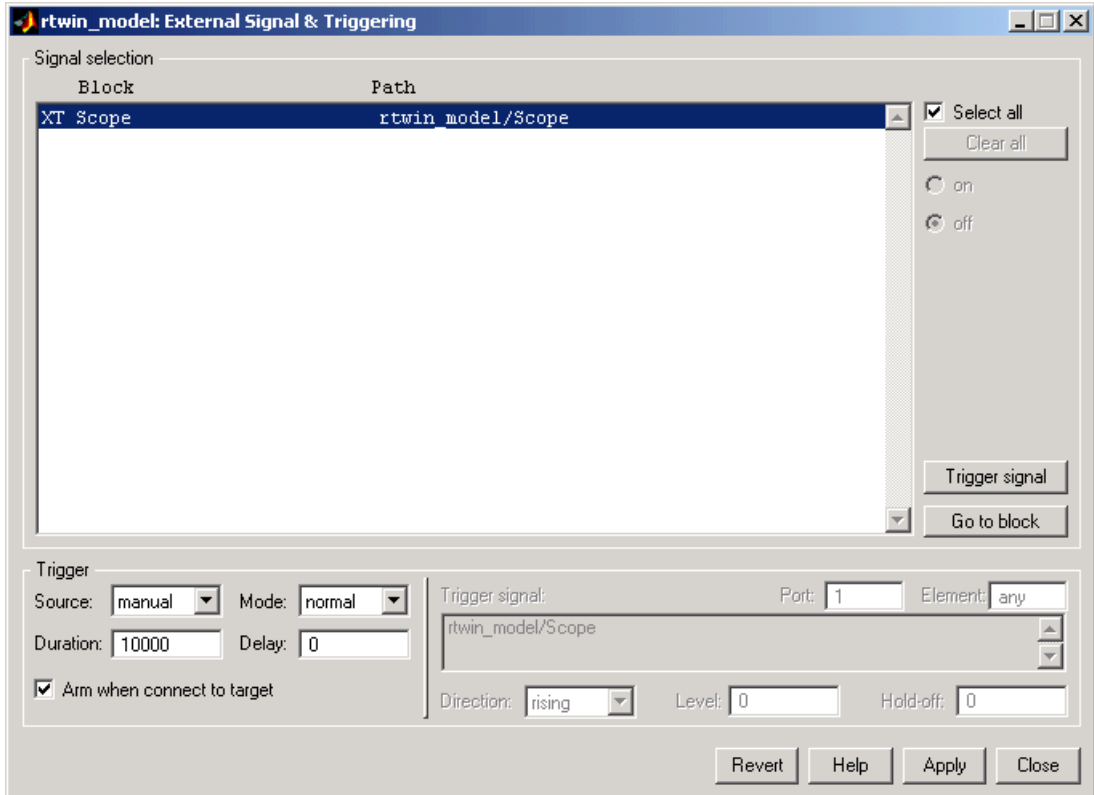
The **Duration** value specifies the number of contiguous points of data to be collected in each buffer of data. We recommend that you enter a **Duration**

value equal to the total number of sample points that you need to collect rather than relying on a series of buffers to be continuous.

If you enter a value less than the total number of sample points, you will lose sample points during the time needed to transfer values from the data buffer to the MATLAB workspace. The Real-Time Windows Target ensures that points are continuous only within one buffer. Between buffers, due to transfer time, some samples will be omitted.

We also recommend setting the time axis for Simulink Scope blocks equal to the sample interval (in seconds) times the number of points in each data buffer. This setting will display one buffer of data across the entire Simulink Scope plot.

The **Signal & Triggering** dialog box will look similar to the figure below.



5 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **Close** to apply the changes to your model and close the **Simulation Parameters** dialog box.

Note You must click the **Apply** or **Close** button on the **Signal and Triggering** dialog box for the changes you made to take effect. Generally it is not necessary to rebuild your real-time application.

Plotting Logged Signal Data

You can use the MATLAB plotting functions for visualizing nonreal-time simulated data or real-time application data.

After running your real-time application and logging data to the MATLAB workspace, you can plot the data. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you saved your data to the variable `ScopeData`:

- 1 In the MATLAB command window, type
`ScopeData`

MATLAB lists the structure of the variable `ScopeData`. The variable `ScopeData` is a MATLAB structure containing the fields `time` vector, `signal` structure, and a string containing the block name.

```
ScopeData =  
    time: [10000x1 double]  
  signals: [1x1 struct]  
  blockName: 'rtwin_model/Scope'
```

To list the contents of the structure `signals`, type

```
ScopeData.signals
```

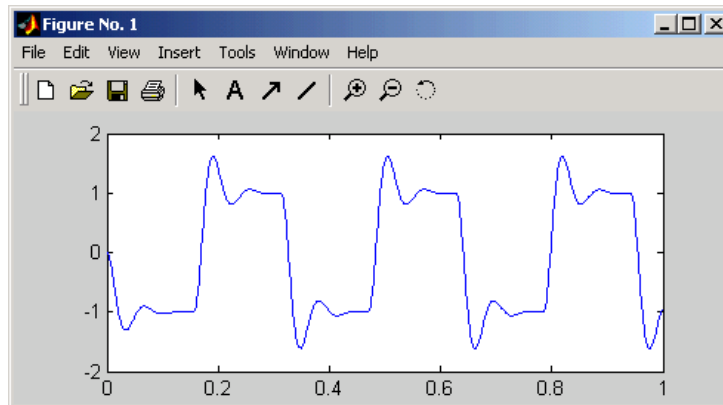
MATLAB lists the structure of the variable `ScopeData.signals`. This structure contains one or more vectors of signal data depending on the number of signal inputs to the Scope block.

```
ans =  
    values: [10000x1 double]  
    label: ''  
    title: ''  
    plotStyle: 1
```

- 2** To plot the first 1000 points, type

```
plot(ScopeData.time(1:1000), ScopeData.signals.values(1:1000))
```

MATLAB plots the first 1000 samples from 0.0000 to 0.9990 seconds.



- 3** The variable `ScopeData` is not automatically saved to your hard disk. To save the variable `ScopeData`, type

```
save ScopeData
```

MATLAB saves the scope data to the file `ScopeData.mat`.

Signal Logging to a Disk Drive

Signal logging is the process of saving (logging) data to a variable in your MATLAB workspace and then saving that data to a MAT-file on your disk drive. This allows you to use MATLAB functions for data analysis and MATLAB plotting functions for visualization. Using the data archiving feature provide in the **External Mode Control Panel**, you can save data to a file during an execution. You cannot save data to a disk drive during a simulation.

To use the data archiving feature with the Real-Time Windows Target, you must add a Scope block to your Simulink model, and you must run an execution of a real-time application.

This section includes the following topics:

- “Entering Scope Parameters” on page 3-29
- “Entering Signal and Triggering Properties” on page 3-31
- “Entering Data Archiving Parameters” on page 3-43
- “Plotting Logged Signal Data” on page 3-35

Simulink external mode does not support data logging with Outport blocks in your Simulink model. This means you do not enter or select parameters on the **Workspace I/O** pane in the **Simulation Parameters** dialog box.

Entering Scope Parameters

Data is saved to a disk drive by first saving the data to the MATLAB workspace through a Simulink Scope block. Scope block parameters need to be set for data to be saved.

After you create a Simulink model and add a Scope block, you can enter the scope parameters for signal logging to a disk drive. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model.

Note If you entered the scope parameters for running a simulation, you may want to look over this procedure because the **Scope parameters** dialog box is related to the **External Signal and Triggering** dialog box and the **Data Archiving** dialog box.

- 1 In the Simulink window, double-click the Scope block.

A Scope window opens.

- 2 On the toolbar, click the **Parameters** button.



A **Scope parameters** dialog box opens.

- 3 Click the **Data history** tab.
- 4 Do one of the following:
 - If you are running a simulation, you can select the **Limit data points to last** check box, and enter the number of sample points to save.
 - If you are running an execution, do not select the select the **Limit data points to last** check box.

Note The **Limit data points to last** check box is related to the **Duration** value in the **External Signal and Triggering** dialog box. The smaller of either value limits the number of sample points saved to the MATLAB workspace. When using the Real-Time Windows Target, we recommend that you use the **Duration** value to set the number of sample points you save.

To set the **Duration** value, see “Entering Signal and Triggering Properties” on page 3-40.

- 5 Select the **Save data to workspace** check box. In the **Variable name** text box, enter the name of a MATLAB variable. The default name is ScopeData.

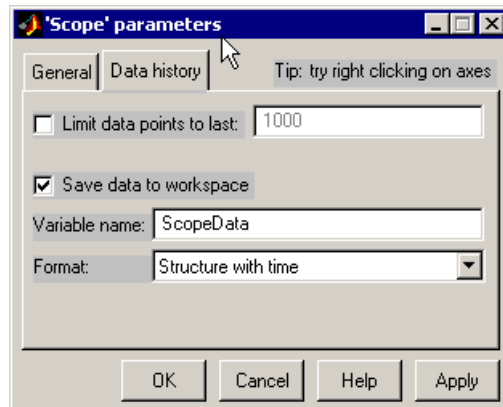
Note The **Scope parameters** dialog box is related to the **Data Archiving** dialog box. In the **Scope parameters** dialog box, you must select the **Save data to workspace** check box to be able to save data to a disk drive for two reasons:

- The data is first transferred from the data buffer to the MATLAB workspace before being written to a MAT-file.
- The **Variable name** entered in the **Scope parameters** dialog box is the same variable in the MATLAB workspace and the variable in the MAT-file.

If you do not select the **Save data to workspace** check box, the MAT-files for data logging will be created, but they will be empty.

- 6 From the **Format** list, choose either Structure with time, Structure, or Array (compatible with V2.0-2.2). For example, to save the sample times and signal values at those times, choose Structure with time.

Your Data history pane will look similar to the figure shown below.



7 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the **Scope parameters** dialog box.

Note When you modify anything in the **Scope parameters** dialog box, you must click the **Apply** or **OK** button for the changes to take effect. Also, you must rebuild your real-time application before connecting and starting a real-time application.

If you do not rebuild, an error dialog box will open. If you do not click **Apply**, your executable will run, but it will use the old settings.

Entering Signal and Triggering Properties

Data is saved to a disk drive by first saving the data to the MATLAB workspace through a Simulink Scope block. Signal and triggering properties need to be set when running a real-time application.

After you create a Simulink model and add a Scope block, you can enter the signal and triggering properties for data logging to a disk drive. This procedure uses the Simulink model `rtwin_model.mdl` as an example and assumes you have already loaded that model:

1 In the Simulink window, and from the **Tools** menu, click **External Mode Control Panel**.

The **External Mode Control Panel** dialog box opens.

2 Click the **Signal & Triggering** button.

The **External Signal & Triggering** dialog box opens.

3 Click the **Select all** button. From the **Source** list, choose `manual`. From the **Mode** list, choose `normal`.

The X under **Signal selection** designates that a signal has been tagged for data collection, and T designates that the signal has been tagged as a trigger signal.

- 4** In the **Duration** box, enter the number of sample points in a data buffer. For example, if you have a sample rate of 1000 samples/second and a stop time of 10 seconds, then enter

10000

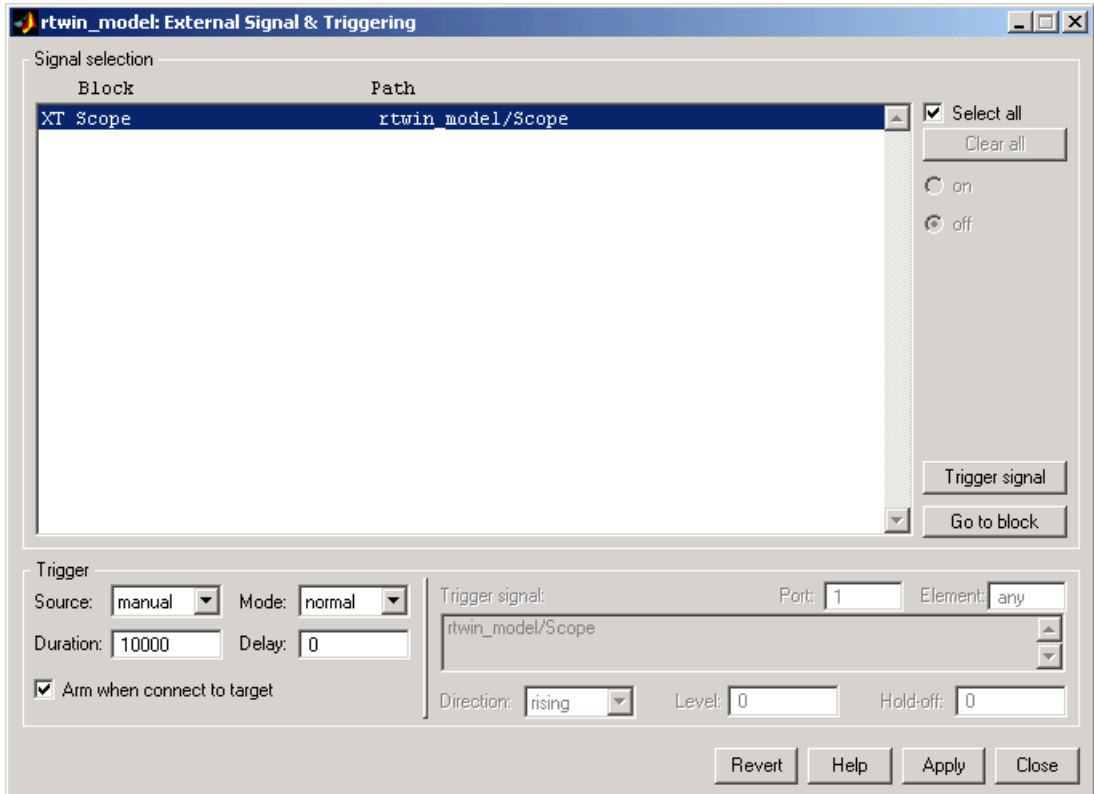
Note The **Duration** value is related to the **Limit data points to last** value in the **Scope parameters** dialog box. The smaller of either value limits the number of sample points saved to the MATLAB workspace. We recommend that you do not select the **Limit data points to last** check box, and use the **Duration** value to set the number of sample points saved.

The **Duration** value specifies the number of contiguous points of data to be collected in each buffer of data. We recommend that you enter a **Duration** value equal to the total number of sample points you need to collect for a run.

If you enter a value much less than the total number of sample points, you may lose logging sample points due to the time needed to transfer values from the data buffer to the MATLAB workspace.

We also recommend setting the time axis for Simulink Scope blocks equal to the sample interval (in seconds) times the number of points in each data buffer. This setting will display one buffer of data across the entire Simulink Scope plot.

The **External Signal & Triggering** dialog box will look similar to the figure shown below.



5 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **Close** to apply the changes to your model and close the **Simulation Parameters** dialog box.

Note You must click the **Apply** or **Close** button on the **Signal and Triggering** dialog box for the changes you made to take effect, but you do not have to rebuild your real-time application.

Entering Data Archiving Parameters

The **Data Archiving** dialog box is related to the **Scope parameters** dialog box. In the **Scope parameters** dialog box, you must select the **Save data to workspace** check box to be able to save data to a disk drive for two reasons:

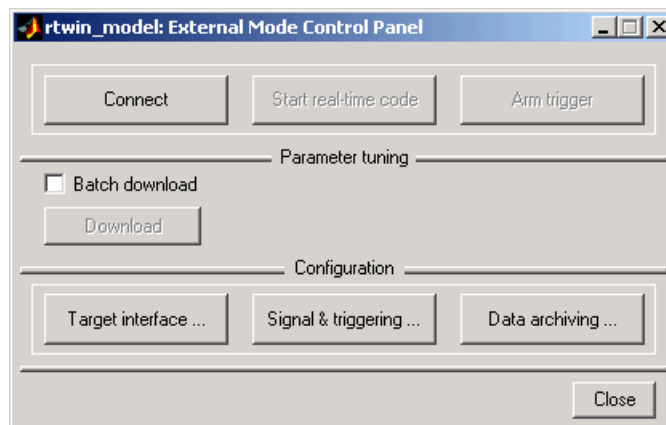
- The data is first transferred from the scope data buffer to the MATLAB workspace before being written to a MAT-file.
- The **Variable name** entered in the **Scope parameters** dialog box is the same variable in the MATLAB workspace and the variable in the MAT-file.

If you do not select the **Save data to workspace** check box in the **Scope parameters** dialog box, the MAT-files for data logging will be created, but they will be empty.

After you create a Simulink model, you can enter the Data Archiving Parameters for data logging to a disk drive:

- 1 In the Simulation window, and from the **Tools** menu, click **External Mode Control Panel**.

The **External Mode Control Panel** dialog box opens.



- 2 Click the **Data archiving** button.

The **External Data Archiving** dialog box opens. This dialog box allows you to specify data archiving options.

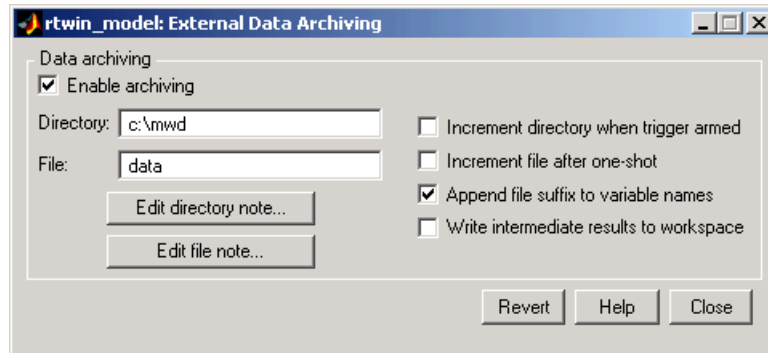
- 3 Select the **Enable archiving** check box.
- 4 In the **Directory** text box, enter the path to a directory on your disk drive. For example, if your MATLAB working directory is named `mwd`, enter
`c:\mwd`
- 5 In the **File** text box, enter the filename prefix for the data files to be saved. For example, enter
`data`

MATLAB names the files `data_0.mat`, `data_1.mat` . . . The Number of files = Total sample points / Duration. For example, if you set the Duration = Total sample points, then only one file is created.

- 6 Select the **Append file suffix to variable names** check box.

Within each MAT-file, a variable is saved with the same name you entered in the **Variable name** text box (Data History pane on the **Scope parameters** dialog box). By selecting the **Append file suffix to variable names** check box, the same suffix that is added to the MAT-file is added to the variable name. For example, if you entered the variable name `ScopeData`, then within the file `data_0.mat` will be a variable `ScopeData_0`.

Your **External Data Archiving** dialog box will look similar to the figure shown below.



- 7 Click the **Close** button.

The parameters you entered are applied to your model.

Note There is no **Apply** button with this dialog box. You must click the **Close** button for the changes you make to take effect.

Plotting Logged Signal Data

You can use the MATLAB plotting functions for visualization of your nonreal-time simulated data or your real-time executed data.

After running your real-time application and logging data to a disk drive, you can plot the data. This procedure uses the Simulink model `rtwin_model.mdl` as an example, and assumes you saved your data to the variable `ScopeData`:

- 1 In the MATLAB command window, type

```
ScopeData
```

MATLAB lists the structure of the variable `ScopeData`. The variable `ScopeData` is a MATLAB structure containing the fields time vector, signal structure, and a string containing the block name.

```
ScopeData =  
    time: [10000x1 double]  
  signals: [1x1 struct]  
  blockName: 'rtwin_model/Scope'
```

- 2 To list the MAT-files saved to your disk drive, type

```
dir *.mat
```

MATLAB displays the MAT-files in your current working directory.

```
data_0.mat
```

- 3 To clear the MATLAB workspace and load the scope data, type

```
clear  
load data_0  
who
```

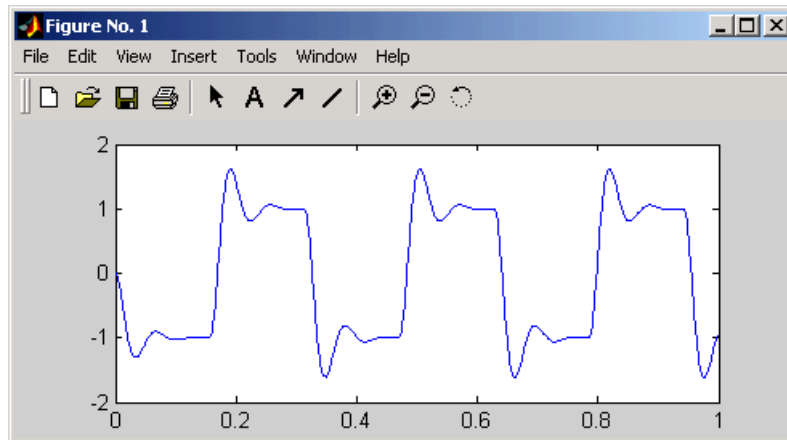
MATLAB displays

```
Your variables are:  
ScopeData_0
```


4 To plot the first 1000 points, type

```
plot(ScopeData_0.time(1:1000),ScopeData_0.signals.values(1:1000))
```

MATLAB plots the first 1000 samples from 0.0000 to 0.9990 second.



Parameter Tuning

Simulink external mode connects your Simulink model to your real-time application. The block diagram becomes a graphical user interface to the real-time application.

This section includes the following topics:

- “Types of Parameters” on page 3-48
- “Changing Model Parameters” on page 3-49

Types of Parameters

You can change parameter values while running the real-time application by changing the values in

- **Block parameters** — Change block parameters by changing the values in the dialog boxes associated with the Simulink blocks. Once you change a value, and click **Apply** or **OK**, the new value immediately replaces the existing parameter while the real-time application continues to run.
- **Block parameters for masked subsystems** — Change block parameters in the user-created dialog boxes associated with a subsystem.
- **MATLAB variables** — Change MATLAB variables by entering the changes through the MATLAB command line, and then press **Ctrl+D** for the changes to be downloaded to your executable. An alternative method to download parameters is to click **Update Diagram** from the **Edit** menu in your Simulink window. Simply changing the value of the MATLAB variable at the MATLAB command line is not sufficient for Simulink to know that the value has changed.

Simulink external mode also supports side-effects functions. For example, given an expression in a Gain block of $2*a+b$, the expression is evaluated and the resulting value is exported to the real-time application during execution.

When a parameter in a Simulink model is changed, the communication module `rtwintext.dll` transfers the data to the external real-time application and changes the model parameters. Only the parameters that do not result in model structure modification can be changed. If the structure is modified, you must recompile the model. Model structure changes are detected automatically using model checksum and reported to the MATLAB command window to avoid conflicts.

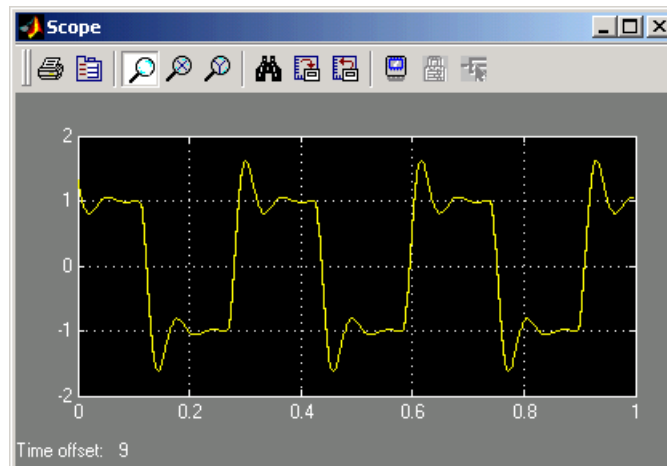
Changing Model Parameters

You must use Simulink external mode to change model parameters. While external mode is running, you can open any Simulink block and change a parameter value. External mode will automatically transfer the new value to the real-time application during execution.

After you start running a your real-time application, you can change parameters and observe the changes to the signals. To start a real-time application, see “Running a Real-Time Application” on page 3-25. This procedure uses the Simulink model `rtwin_model.mdl` as an example. It assumes you have created a real-time application and are running an execution:

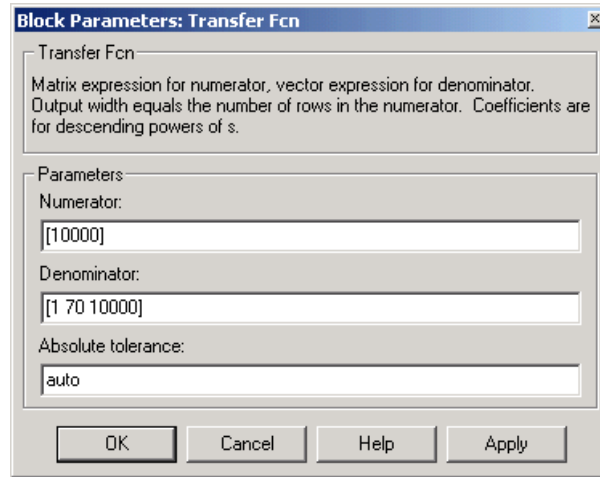
- 1 From the **Simulation** menu, click **Start Real-Time** code.

The real-time execution starts running and signal data is displayed in the Scope window.

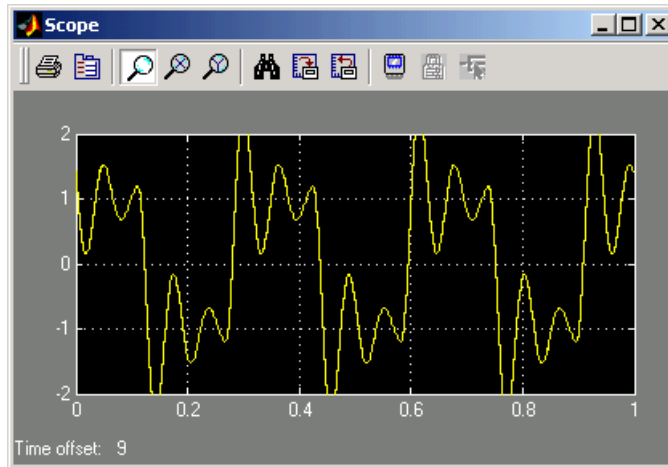


- 2 From the Simulink block diagram, click the Transfer Function block.

The **Block Parameters: Transfer Fcn** dialog box opens.



- 3 In the **Denominator** box, change 70 to 30. Click **OK**. The effect of changing a block parameter is shown in the Scope window.



Advanced Procedures

I/O Boards	4-3
I/O Board Dialog Box	4-3
ISA Bus Board	4-5
PCI Bus Board	4-6
PC/104 Board	4-6
Compact PCI Board	4-6
PCMCIA Board	4-6
I/O Driver Blocks	4-8
Real-Time Windows Target Library	4-8
Simulink Library	4-10
Analog Input Block	4-12
Analog Output Block	4-14
Digital Input Block	4-17
Digital Output Block	4-19
Counter Input Block	4-22
Encoder Input Block	4-25
Output Signals from an I/O Block	4-27
Variations with Channel Selection	4-29
Using Analog I/O Drivers	4-32
I/O Driver Characteristics	4-32
Normalized Scaling for Analog Inputs	4-33

The Real-Time Windows Target provides driver blocks for more than 200 I/O boards. These driver blocks connect the physical world to your real-time application.

This chapter includes the following sections:

- **I/O Boards** — Install I/O boards and enter hardware information
- **I/O Driver Blocks** — Select analog and digital driver blocks from the Simulink library and add to your Simulink model
- **Using Analog I/O Drivers** — Convert normalized I/O signals to more meaningful model parameters

I/O Boards

Typically I/O boards are preset from the factory for certain base addresses, voltage levels, and unipolar or bipolar modes of operation. Boards often include switches or jumpers that allow you to change many of these initial settings. For information about setting up and installing any I/O board, read the board manufacturer's documentation.

This section includes the following topics:

- **I/O Board Dialog Box** — Select the physical boards installed in your computer, and enter board settings
- **ISA Bus Board** — Enter base address
- **PCI Bus Board** — Enter or determine slot number and install drivers from the board manufacturer
- **PC/104 Board** — Enter base address
- **Compact PCI Board** — Enter or determine slot number and install drivers from the board manufacturer
- **PCMCIA Board** — Install drivers from the board manufacturer

I/O Board Dialog Box

Usually, the drivers with the Real-Time Windows Target provide the same flexibility of settings offered by the board manufacturer. You enter the I/O board settings in the I/O board dialog box. There are three types of settings:

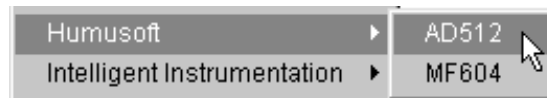
- **Software selectable** — Select check boxes in the I/O board dialog box. The driver writes the settings you selected to the board. Examples include A/D gain inputs, and selecting unipolar or bipolar D/A outputs.
- **Jumper selectable and software readable** — Set jumpers or switches on the physical board. The driver reads the settings you selected.
- **Jumper selectable, but not software readable** — Set jumpers or switches on the physical board, and then manually enter the same settings in the I/O board dialog box. These entries must match the hardware switches or jumpers you set on the board. This is necessary because some manufacturers do not provide a means for the I/O driver to read all of the board settings with software. Examples include base address, D/A gain, and differential or single-ended A/D inputs.

After you add an I/O driver block to your Simulink model, you can select and configure the I/O board installed in your computer. This procedure uses the AD512 I/O board from Humusoft as an example:

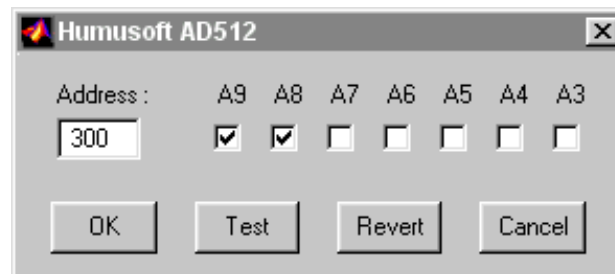
- 1 Double-click an I/O driver block.

The **Block Parameters** dialog box opens.

- 2 Click the **Install new board** button. From the list, point to a manufacture, and then click a board name. For example, point to Humusoft, and then click AD512.



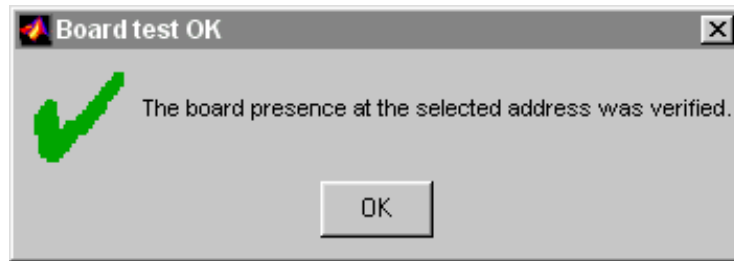
The I/O board dialog box opens. The name of this dialog box depends on which I/O board you selected. The dialog box for the Humusoft AD512 board is shown below.



- 3 Select one of the following:
 - For an ISA bus board, enter a base address. This value must match the base address switches or jumpers set on the physical board. For example, to enter a base address of 0x300 in the address box, type
300
You can also select the base address by selecting check boxes A9 through A3.
 - For a PCI bus board, enter the **PCI slot** or select the **Auto-detect** check box.

- 4 Click the **Test** button.

The Real-Time Windows Target tries to connect to the selected board, and if successful, displays the following message.



- 5 On the message box, click **OK**, and then on the I/O board dialog box, click **OK** again.

The I/O driver **Block Parameters** dialog box closes, and the parameter values are saved with your Simulink model.

The I/O board information is included with each I/O driver block. You only install and enter the board information once with the first I/O driver block you add to your model. When you add another I/O driver block, choose the I/O board from the list of installed boards. You do not need to enter any board information.

ISA Bus Board

Most ISA bus I/O boards are preset with a base address of 0x300. If you are using multiple I/O boards or other boards (for example, network cards) that already use the address 0x300, you must set your board with another base address.

In the I/O board dialog box, enter the same base address that you set on the physical board. You open the I/O board dialog box from any I/O driver **Block Parameters** dialog box.

PCI Bus Board

You do not have to set a base address with a PCI board.

The plug-and-play feature of Microsoft Windows assigns a PCI slot number. You can enter the slot number into the I/O board dialog box, or you can let the driver determine the slot number for you. You open the I/O board dialog box from any I/O driver **Block Parameters** dialog box.

We recommend that before you use a PCI or PCMCIA board, you install the drivers supplied by the board manufacturer. The Real-Time Windows Target does not use these manufacturer supplied drivers. However, they sometimes initiate the plug-and-play recognition of the board. Without these drivers installed, the board may be invisible to your computer and Real-Time Windows Target.

PC/104 Board

Most PC/104 bus I/O boards are preset with a base address of 0x300. If you are using multiple I/O boards or other boards (for example, network cards) that already use the address 0x300, you must set your board with another base address.

In the I/O board dialog box, enter the same base address that you set on the physical board. You open the I/O board dialog box from any I/O driver **Block Parameters** dialog box.

Compact PCI Board

Using a compact-PCI board requires that you use a compact PC (industrial PC). In addition, you need to install Windows, MATLAB, Simulink, the Real-Time Windows Target, and a C compiler on the compact PC.

PCMCIA Board

The plug-and-play feature of Microsoft Windows assigns a base address automatically. You can enter this address into the I/O board dialog box, or you can let the driver determine the address for you. You open the I/O board dialog box from any I/O driver **Block Parameters** dialog box.

We recommend that before you use a PCI or PCMCIA board, you install the drivers supplied by the board manufacturer. The Real-Time Windows Target does not use these manufacture supplied drivers. However, they sometimes

initiate the plug-and-play recognition of the board. Without these drivers installed, the board may be invisible to your computer and Real-Time Windows Target.

I/O Driver Blocks

The Analog Input, Analog Output, Digital Input, and Digital Output blocks provide an interface to your physical I/O boards and your real-time application. They ensure that the C code generated with Real-Time Workshop correctly maps block diagram signals to the appropriate I/O channels.

You can have multiple blocks associated with each type of I/O block and board. For example, you can have one Analog Input block for channels 1-4 and another block for channels 5-8.

This section includes the following topics:

- **Real-Time Windows Target Library** — Add an Analog Input block to your Simulink model from the Real-Time Windows Target block library
- **Simulink Library** — Add an Analog Input block to your Simulink model from the Simulink block library
- **Analog Input Block** — Select analog input channels and voltage range
- **Analog Output Block** — Select analog output channels, voltage range, initial values, and final values
- **Digital Input Block** — Select digital lines or channels
- **Digital Output Block** — Select digital lines or channels, initial values, and final values
- **Counter Input Block** — Select and connect specific counter input channels to your Simulink model
- **Encoder Input Block** — Select and connect specific encoder input channels to your Simulink model

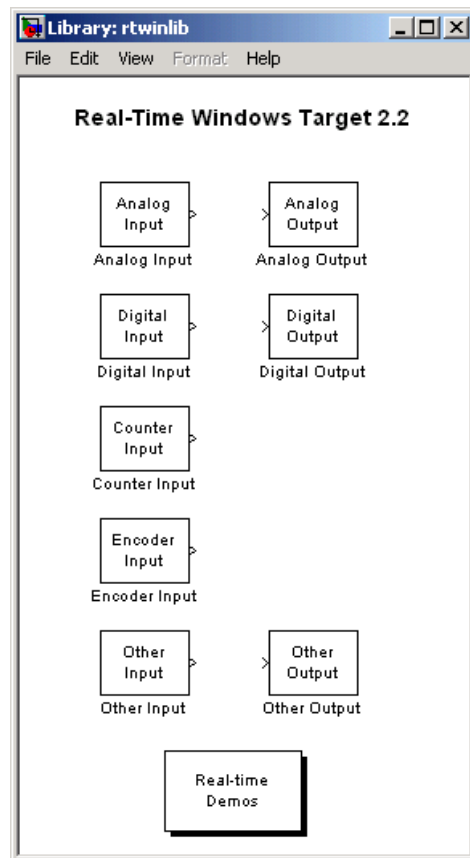
Real-Time Windows Target Library

The Real-Time Windows Target I/O driver blocks allow you to select and connect specific analog channels and digital lines to your Simulink model through I/O driver blocks.

After you create a Simulink model, you can add an I/O block. This procedure adds an Analog Input block and uses the Simulink model `rtwin_model.mdl` as an example:

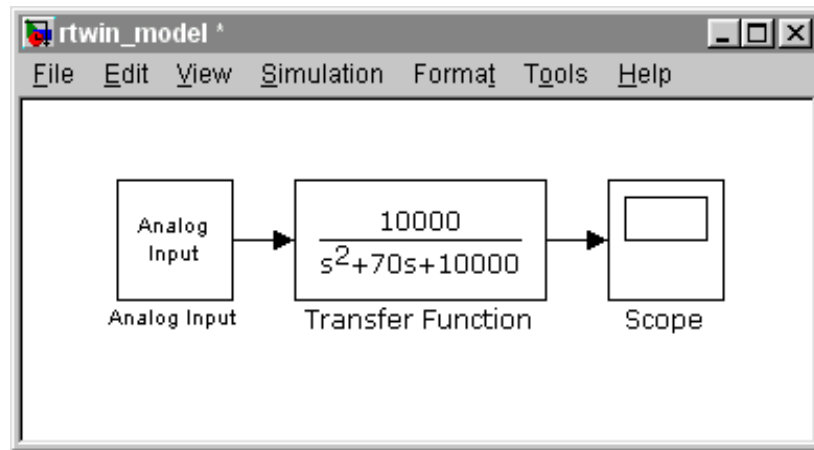
- 1 In the MATLAB command window, type
`rtwinlib`

The Real-Time Windows Target block library window opens.



- 2 Click-and drag the Analog Input block to your Simulink model. Connect the Analog Input block to the Transfer Function block.

Your Simulink model will look similar to the following figure.



Your next task is to enter parameters for the Analog Input block. See “Analog Input Block” on page 4-12.

Simulink Library

The Real-Time Windows Target I/O driver blocks allow you to select and connect specific analog channels and digital lines to your Simulink model through I/O driver blocks.

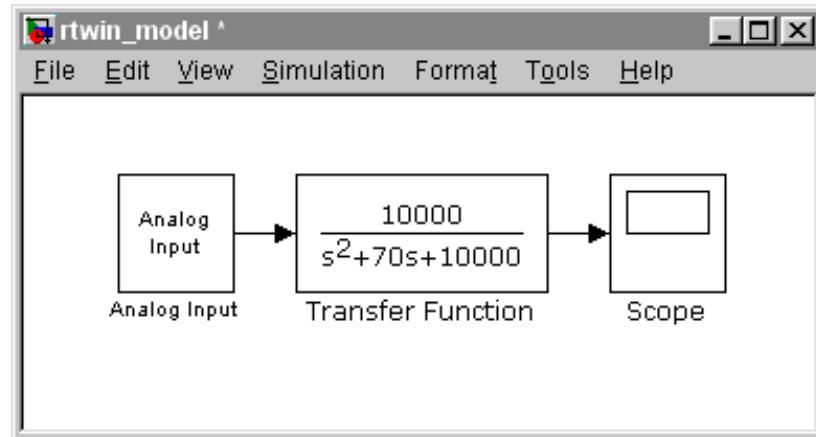
After you create a Simulink model, you can add an I/O block. This procedure adds an Analog Input block and uses the Simulink model `rtwin_model.mdl` as an example:

- 1 In the Simulink window, and from the **View** menu, click **Show Library Browser**.

The Simulink Library Browser opens.

- 2 In the left column, double-click **Real-Time Windows Target**. Click and drag the Analog Input block to your Simulink model. Connect the Analog Input block to the Transfer Function block.

Your Simulink model will look similar to the figure show below.



Your next task is to enter parameters for the Analog Input block. See “Analog Input Block” on page 4-12.

Analog Input Block

The Real-Time Windows Target I/O blocks allow you to select and connect specific analog channels to your Simulink model.

After you add an Analog Input block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's AD512 I/O board as an example:

- 1 Double-click the Analog Input block.

The **Block Parameters: Analog Input** dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the **Simulation Parameters** dialog box. For example, enter

0.001

- 3 In the **Input channels** box, enter a channel vector that selects the analog input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all eight analog input channels on the AD512 board, enter

[1,2,3,4,5,6,7,8] or [1:8]

If you want to use the first three analog input channels, enter

[1,2,3]

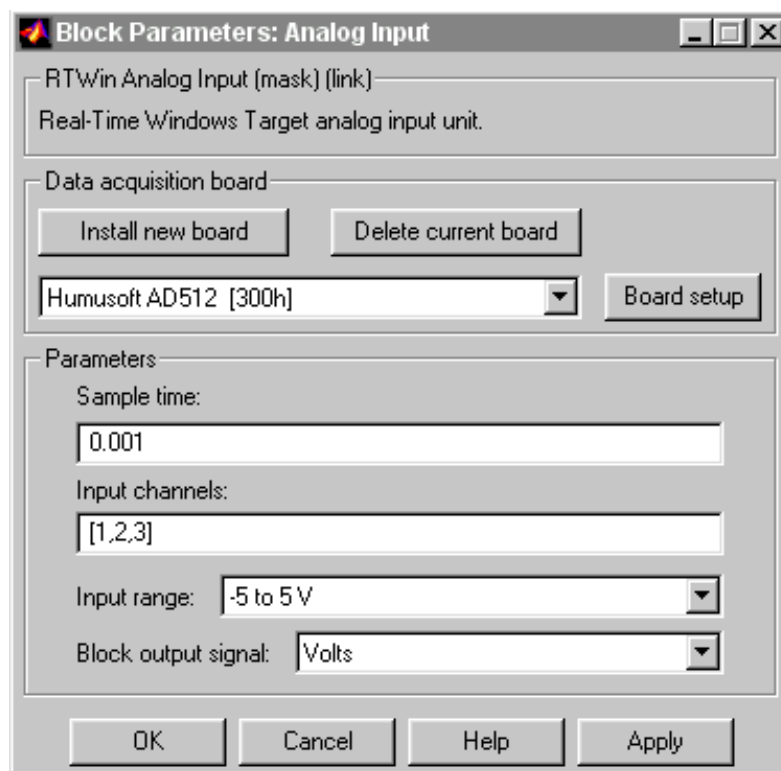
- 4 From the **Input range** list, choose the input range for all of the analog input channels you entered in the Input channels box. For example, with the AD512 board, choose -5 to 5 V.

Note If you want the input range to be different for different analog channels, you need to add a I/O block for each different input range.

- 5 From the **Block output signal** list, choose from the following options:
 - Volts — Returns a value equal to the analog voltage
 - Normalized unipolar — Returns a full range value of 0 to +1 regardless of the input voltage range. For example, an analog input range of 0 to +5 volts and -5 to +5 volts would both be converted to 0 to +1.
 - Normalized bipolar — Returns a full range value of -1 to +1 regardless of the input voltage range.

- **Raw** — Returns a value of 0 to $2^n - 1$. For example, a 12-bit A/D converter would return values of 0 to $2^{12} - 1$ (0 to 4095). The advantage of this method is the returned value is always an integer with no round-off errors.

If you chose **Volts**, your dialog box will look similar to the figure shown below.



- 6 Select one of the following:
 - Click the **Apply** button to apply the changes to your model and leave the dialog box open.
 - Click the **OK** button to apply the changes to your model and close the **Block Parameters: Analog Input** dialog box.

Analog Output Block

The Real-Time Windows Target I/O blocks allow you to select and connect specific analog channels to your Simulink model.

After you add an Analog Output block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's AD512 I/O board as an example:

- 1 Double-click the Analog Output block.

The **Block Parameters: Analog Output** dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the **Simulation Parameters** dialog box. For example, enter

0.001

- 3 In the **Output channels** box, enter a channel vector that selects the analog input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select both analog output channels on the AD512 board, enter

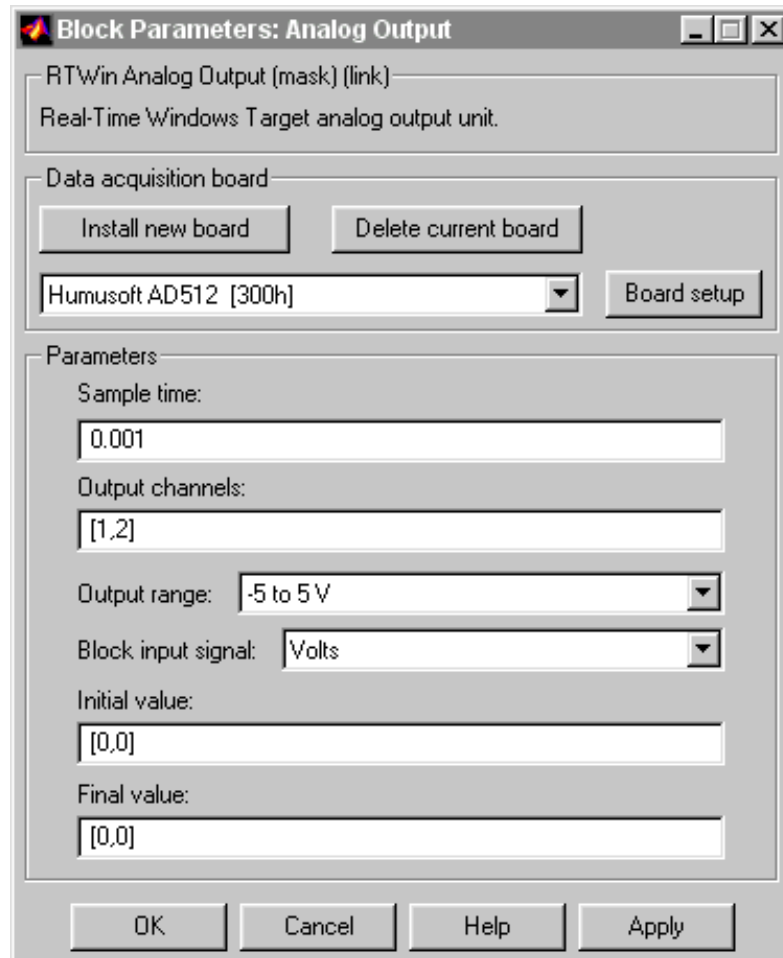
[1,2] or [1:2]

- 4 From the **Output range** list, choose the input range for all of the analog input channels you entered in the Input channels box. For example, with the AD512 board, choose -5 to 5 V.

Note If you want the input range to be different for different analog channels, you need to add a I/O block for each different input range.

- 5 From the **Block input signal** list, choose from the following options:
 - Volts — Expects a value equal to the analog output voltage
 - Normalized unipolar — Expects a value between 0 to +1 that is converted to the full range of the output voltage regardless of the output voltage range. For example, an analog output range of 0 to +5 volts and -5 to +5 volts would both be converted from values between 0 and +1.
 - Normalized bipolar — Expects a value between -1 to +1 that is converted to the full range of the output voltage regardless of the output voltage range.
 - Raw — Expects a value of 0 to $2^n - 1$. For example, a 12-bit A/D converter would expect a value between 0 and $2^{12} - 1$ (0 to 4095). The advantage of this method is the expected value is always an integer with no round-off errors.
- 6 Enter the initial value for each analog output channels you entered in the **Output channels** box. For example, if you entered [1, 2] in the **Output channels** box, and you want an initial value of 0 volts, enter [0, 0].
- 7 Enter a final value for each analog channel you entered in the **Output channels** box. For example, if you entered [1, 2] in the **Output channels** box, and you want final values of 0 volts, enter [0, 0].

If you chose Volts, your dialog box will look similar to the figure shown below.



- 8 Do one of the following:
 - Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the **Block Parameters: Analog Output** dialog box.

Digital Input Block

The Real-Time Windows Target I/O blocks allow you to select and connect specific digital lines or digital channels to your Simulink model.

After you have added and Digital Input block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's AD512 I/O board as an example:

- 1 Double-click the Digital Input block.

The **Block Parameters: Digital Input** dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the **Simulation Parameters** dialog box. For example, enter

0.001

- 3 In the **Input channels** box, enter a channel vector that selects the digital input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all eight digital input channels on the AD512 board, enter

[1,2,3,4,5,6,7,8] or [1:8]

If you want to use the first four digital input lines, enter

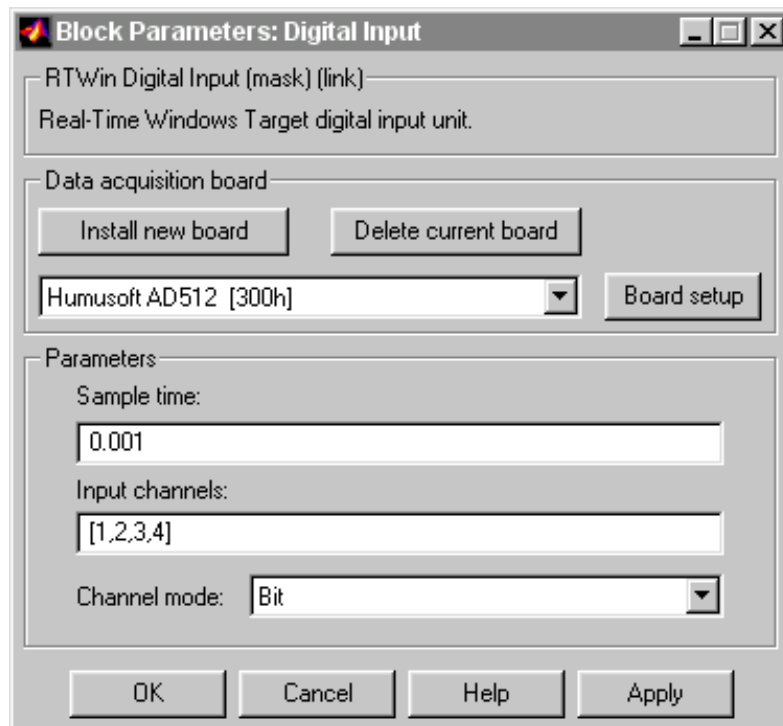
[1,2,3,4]

If you have one 8-bit digital channel, enter [1]. If you have two 8-bit digital channels, enter [1 9], and from the **Channels mode** list, choose Byte.

4 From the Channel mode list, choose one of the following options:

- Bit — Returns a value of 0 or 1
- Byte — Groups eight digital lines into one digital channel and returns a value of 0 to 255.

If you chose Bit, your dialog box will look similar to the figure shown below.



5 Do one of the following:

- Click **Apply** to apply the changes to your model and leave the dialog box open.
- Click **OK** to apply the changes to your model and close the **Block Parameters: Digital Input** dialog box.

Digital Output Block

The Real-Time Windows Target I/O blocks allow you to select and connect specific digital lines or digital channels to your Simulink model.

After you have added and Digital Output block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's AD512 I/O board as an example:

- 1 Double-click the Digital Output block.

The **Block Parameters: Digital Output** dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the **Simulation Parameters** dialog box. For example, enter

0.001

- 3 In the **Output channels** box, enter a channel vector that selects the analog input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all eight analog input channels on the AD512 board, enter

[1,2,3,4,5,6,7,8] or [1:8]

If you want to use the first four digital input lines, enter

[1,2,3,4]

If you have one 8-bit digital channel, enter [1]. If you have two 8-bit digital channels, enter [1 9], and from the **Channel mode** list, choose Byte.

- 4 From the **Channel mode** list, choose from one of the following:
 - Bit — Expects a value of 0 or 1
 - Byte — Expects a value of 0 to 255 that is converted to one digital channel of eight digital lines

- 5** Enter the initial values for each digital output line or channel you entered in the **Output channels** box. For example, if you entered [1, 2, 3, 4] in the **Output channels** box, and you want initial values of 0 and 1, enter

[0,0,1,1]

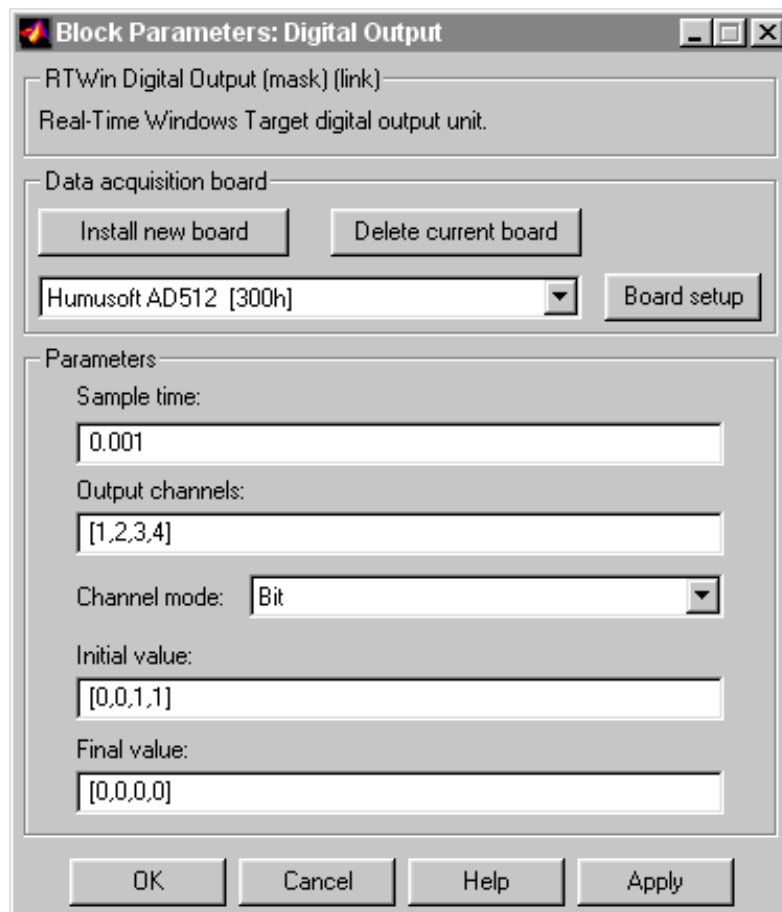
If from the **Channel mode** list, you choose **Byte**, then enter a value between 0 and 255 for each digital output channel. For example, for one byte (8 digital lines) with an initial value of 25, enter [25]. For two bytes (16 digital lines) with initial values of 25 and 50, enter [25 50].

- 6** Enter a final value for each digital output channel you entered in the **Output channels** box. For example, if you entered [1, 2, 3, 4] in the **Output channels** box, and you want final values of 0, enter

[0,0,0,0]

If from the **Channel mode** list, you choose **Byte**, then enter a value between 0 and 255 for each digital output channel.

If you chose Bit, your dialog box will look similar to the figure shown below.



- 7 Do one of the following:
 - Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the **Block Parameters: Digital Output** dialog box.

Counter Input Block

This Real-Time Windows Target I/O blocks allow you to select and connect specific counter input channels to your Simulink model.

After you have added a Counter Input block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's MF604 I/O board as an example:

- 1 Double-click the Counter Input block.

The **Block Parameters: Counter Input** dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the **Simulation Parameters** dialog box. For example, enter

0.001

- 3 In the **Input channels** box, enter a channel vector that selects the counter input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all 4 counter input channels on the MF604 board, enter

[1,2,3,4] or [1:4]

- 4** From **Reset after read**, which determines if the counter should be reset to zero after its value has been read, choose one of the following options:
- **never** — Do not reset after reading.
 - **always** — Always reset after reading.
 - **level** — Reset after reading if block input is nonzero. This will add an input to the Counter Input block.
 - **rising edge** — Reset after reading if block input changes from zero to nonzero between the last two successive readings. This will add an input to the Counter Input block.
 - **falling edge** — Reset after reading if the block input changes from nonzero to zero between last two successive readings. This will add an input to the Counter Input block.
 - **either edge** — Reset after reading if the block input changes either from zero to nonzero or from nonzero to zero between the last two successive readings. This will add an input to the Counter Input block.
- 5** From **Clock input active edge**, which determines which clock edge should increment the counter, select:
- **rising** — Low to high transitions
 - **falling** — High to low transitions

Not all counter chips support selecting the input edge. In this case, the pull-down menu will reflect the supported option only.

- 6** From **Gate input functionality**, which defines the action of the counter's gate pin, select:
- none — Gate is disabled.
 - enable when high — Counting is disabled when the gate is low and enabled when the gate is high.
 - enable when low — Counting is disabled when the gate is high and enabled when the gate is low.
 - start on rising edge — Counting is disabled until low to high transition of the gate occurs.
 - start on falling edge — Counting is disabled until high to low transition of the gate occurs.
 - reset on rising edge — Counter is reset when low to high transition of the gate occurs.
 - reset on falling edge — Counter is reset when high to low transition of the gate occurs.
 - latch on rising edge — The count of the counter is remembered when low to high transition of the gate occurs.
 - latch on falling edge — The count of the counter is remembered when high to low transition of the gate occurs.
 - latch & reset on rising edge — The count of the counter is remembered and then the counter is reset when low to high transition of the gate occurs.
 - latch & reset on falling edge — The count of the counter is remembered and then the counter is reset when high to low transition of the gate occurs.

Not all counter chips support all gate modes. Only supported gate modes are shown in the pull-down menu.

- 7** Do one of the following:
- Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the **Block Parameters: Counter Input** dialog box.

Encoder Input Block

This Real-Time Windows Target I/O blocks allow you to select and connect specific encoder input channels to your Simulink model.

After you have added an Encoder Input block to your Simulink model, you can enter the parameters for this I/O driver. This procedure uses Humusoft's MF604 I/O board as an example:

- 1 Double-click the Encoder Input block.

The **Block Parameters: Encoder Input** dialog box opens.

- 2 In the **Sample time** box, enter the same value you entered in the **Fixed step size** box from the **Simulation Parameters** dialog box. For example, enter
0.001
- 3 In the **Input channels** box, enter a channel vector that selects the encoder input channels you are using on this board. The vector can be any valid MATLAB vector form. For example, to select all 4 encoder input channels on the MF604 board, enter
[1,2,3,4] or [1:4]

- 4** Encoders typically use two sets of stripes, shifted in phase, to optically detect the amplitude and direction of movement. The **Quadrature mode** parameter specifies which encoder stripe edges should be counted.
- **double** — Counts the rising edges from both stripe sets
 - **single** — Counts the rising edges from one stripe set
 - **quadruple** — Counts rising and falling edges from both stripe sets

Quadruple mode yields 4 times more pulses per revolution than the single mode. Therefore, quadruple is more precise and is recommended unless other parameters dictate otherwise.

- 5** The encoder interface chip has a reset pin in addition to encoder inputs. This pin is usually connected to the index output of the encoder. However, it can be connected to any signal or may not be used at all. The **Reset input function** specifies the function of this pin.
- **gate** — Enables encoder counting
 - **reset** — Level reset of the encoder count
 - **rising edge index** — Resets the encoder count on the rising edge
 - **falling edge index** — Resets the encoder count on the falling edge
- 6** The encoder interface chip has a built-in lowpass filter that attempts to filter out any high frequencies which are interpreted as noise. The **Input filter clock frequency** is the cutoff frequency (Hz) of this filter. The cutoff frequency you specify is rounded to the nearest frequency supported by the chip.

If the encoder is moving slowly and high frequency noise is present, employ the filter to eliminate the noise. This keeps the noise from being counted as encoder pulses. If the encoder is moving quickly, the filter can filter out all of the high frequency pulses including those you want to count. In this case, consider leaving the filter disabled by setting the cutoff frequency to Inf.

- 7** Do one of the following:
- Click **Apply** to apply the changes to your model and leave the dialog box open.
 - Click **OK** to apply the changes to your model and close the **Block Parameters: Digital Input** dialog box.

Output Signals from an I/O Block

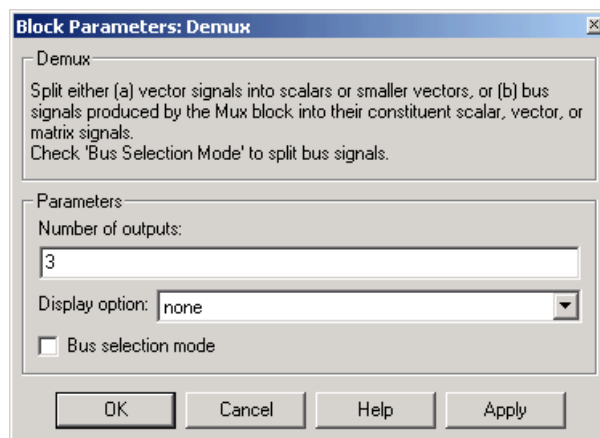
I/O driver blocks output multiple signals as a vector instead of individual channels or lines. To connect the individual channels and lines to parts of your Simulink model, you need to separate the vector with a **Demux** block.

After you add and configure an I/O driver block in your Simulink model, you can separate and connect the output signals from the blocks:

- 1 In the Simulink window, and from the **View** menu, click **Show Library Browser**.

The Simulink Library Browser opens.

- 2 Double-click **Signal Routing**. From the list in the right column, click-and-drag **Demux** to your Simulink model.
- 3 Double-click the Demux block. The **Block Parameters: Demux** dialog box opens. Enter the number of lines leaving the Demux block. For example, if you entered three channels in the Analog Input driver block, enter 3 in the **Number of outputs** box.

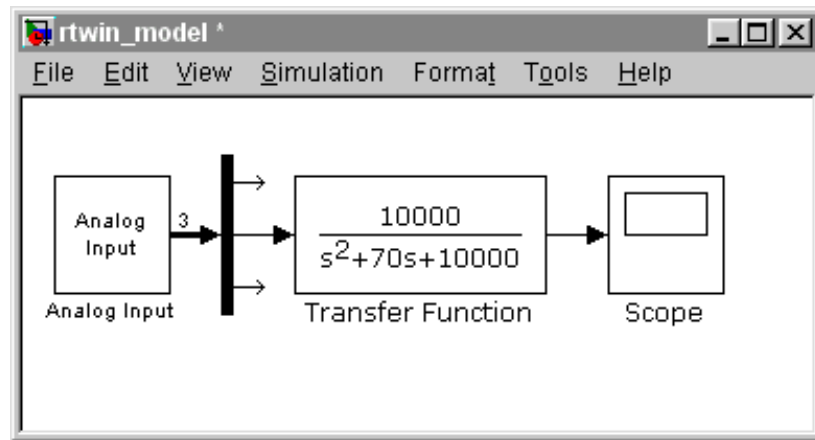


- 4 Click **OK**.

5 Finish making connections and selecting display options.

- Connect the Analog Input block to the Demux block input.
- Connect each of the Demux block output lines to the input of other blocks.
- In the Simulink window, and from the **Format** menu, click **Wide Nonscalar Lines**, and click **Signal Dimensions**.

Your model will look similar to the figure shown below. In this simple example, inputs 1 and 2 are not connected, but they could be connected to other Simulink blocks.



Variations with Channel Selection

For a better understanding of how to specify device settings when using both analog and digital signals, this section uses the I/O board DAS-1601 from Keithley-Metrabyte as an example. The following is a specification summary of the DAS-1601 board:

- Analog input (A/D) — 16 single-ended or 8 differential analog inputs (12-bit), polarity is switch configured as either unipolar (0 to 10 volts) or bipolar (+/- 10 volts). Gain is software configured to 1, 10, 100, and 500.
- Digital input — Four unidirectional digital inputs
- Analog output (D/A) — Two analog outputs (12-bit). Gain is switch configured as 0 to 5 volts, 0 to 10 volts, +/- 5 volts, or +/- 10 volts
- Digital output — Four unidirectional digital outputs
- Base address — Switch configured base address

This section explores different configurations for input signals.

Once an Analog Input block has been placed in the model and the I/O board selected and configured, you can set up the Analog Input block to handle input signals.

Single analog input — The most basic case is for a single analog input signal that will be physically connected to the first analog input channel on the board. In the **Block Parameter: Analog Input** dialog box, and the **Input channels** box, enter

1 or [1]

The use of brackets is optional for a single input.

Input vector with differential analog — Analog channels are numbered starting with channel 1 and continue until you reach a number corresponding to the maximum number of analog signals supported by the I/O board.

In the case of the DAS-1601, when configured as differential inputs, eight analog channels are supported. The analog input lines are numbered 1 through 8. The complete input vector is

[1 2 3 4 5 6 7 8] or [1:8]

If you wanted to use the first four differential analog channels, enter

[1 2 3 4]

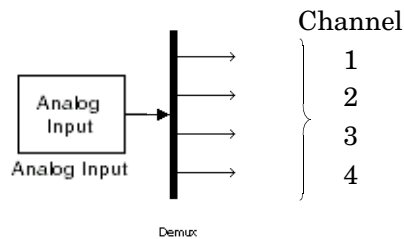
Input vector with single-ended analog — Now, assume your DAS-1601 board is configured to be single-ended analog input. In this case, 16 analog input channels are supported. The complete input vector is

[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16] or [1:16]

To use the first four single-ended analog input channels, enter

[1 2 3 4] or [1:4]

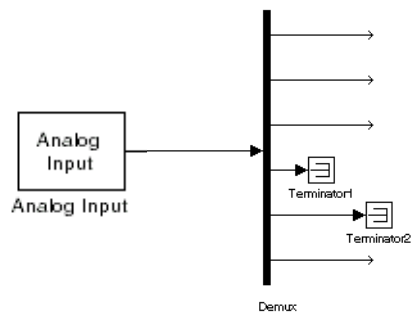
This illustration shows the resulting block diagram.



We do not recommend specifying more channels than you actually use in your block diagram. This results in additional overhead for the processor with A/D or D/A conversions. In this case, for example, even though some channels are not actually used in the block diagram, these channels are still converted.

You could attach terminator blocks to channels 4 and 5 inside your block diagram after passing the Analog Input block vector into a Demux block. Adding terminator blocks provides you with graphical information in your block diagram to clearly indicate which channels you connected or are available. The penalty is that even the terminated channels are converted, adding some computational overhead.

This illustration shows the block implementation.



Depending on the board and the number of channels used, I/O conversion time can affect the maximum sample rate that can be achieved on your system. Rather than converting unused channels, we recommend specifying only the set of channels that are actually needed for your model.

Using Analog I/O Drivers

Control systems have unique requirements for I/O devices that the Real-Time Windows Target supports.

This section includes the following topics:

- “I/O Driver Characteristics” on page 4-32
- “Normalized Scaling for Analog Inputs” on page 4-33

I/O Driver Characteristics

The Real-Time Windows Target uses off-the-shelf I/O boards provided by many hardware vendors. These boards are often used for data acquisition independently of the Real-Time Windows Target. In such environments, board manufacturers usually provide their own I/O device drivers for data acquisition purposes. This use differs significantly from the behavior of drivers provided with the Real-Time Windows Target.

In data acquisition applications, data is often collected in a burst or frame consisting of many points, perhaps 1,000 or possibly more. The burst of data becomes available once the final point is available. This approach is not suitable for use in automatic control applications since it results in latencies equal to $1000 * T_{\text{sample}}$ for each point of data.

In contrast, drivers used by the Real-Time Windows Target capture a single point of data at each sample interval and considerable effort is made to minimize the latency between collecting a data point and using the data in the control system algorithm. This is the reason why a board that specifies a maximum sample rate (for data acquisition) may be stated to achieve sample rates well in excess of the rates that are achievable in the Real-Time Windows Target. For data acquisition, such boards are usually acquiring data in bursts and not in a point-by-point fashion which is more appropriate for stable control systems.

Normalized Scaling for Analog Inputs

The Real-Time Windows Target allows you to normalize I/O signals internal to the block diagram. Generally, inputs represent real-world values such as angular velocity, position, temperature, pressure, and so on. This ability to choose normalized signals allows you to

- Apply your own scale factors
- Work with meaningful units without having to convert from voltages

When using an Analog Input block, you select the range of the external voltages that are received by the board, and you choose the block output signal. For example, the voltage range could be set to 0 to +5 V, and the block output signal could be chosen as Normalized unipolar, Normalized bipolar, Volts, or Raw.

If you prefer to work with units of voltage within your Simulink block diagram, you can choose Volts.

If you prefer to apply your own scaling factor, you can choose Normalized unipolar or Normalized bipolar, add a Gain block, and add an offset to convert to a meaningful value in your model.

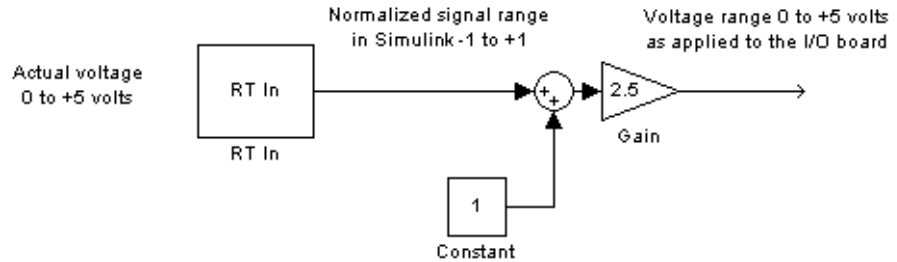
If you prefer unrounded integer values from the analog-to-digital conversion process, you can choose Raw.

Choose 0 to +5 Volts and Normalized Bipolar

From the Input range list, choose 0 to +5 V, and from the **Block output signal** list, choose Normalized bipolar. This example converts a normalized bipolar value to volts, but you could also easily convert directly to another parameter in your model.

0 to 5 volts --> $([-1 \text{ to } 1] \text{ normalized} + 1) * 2.5$

In your block diagram, you can do this as follows.

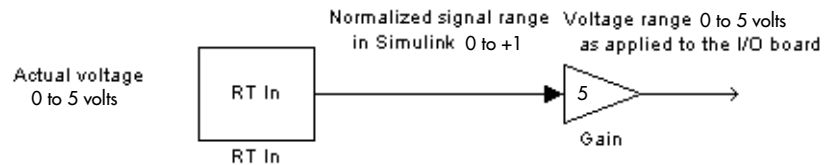


Choose 0 to +5 Volts and Normalized Unipolar

From the **Input range** list, choose 0 to +5 V, and from the **Block output signal** list, choose Normalized unipolar. This example converts a normalized unipolar value to volts, but you could also easily convert directly to another parameter in your model.

$$0 \text{ to } 5 \text{ volts} \rightarrow ([0 \text{ to } 1] \text{ normalized} * 5.0)$$

In your block diagram, you can do this as follows.

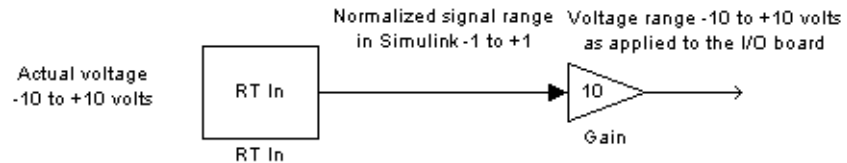


Choose -10 to +10 Volts and Normalized Bipolar

From the Input range list, choose -10 to +10 V, and from the Block output signal list, choose Normalized bipolar. This example converts a normalized bipolar value to volts, but you could also easily convert directly to another parameter in your model.

$$-10 \text{ to } 10 \text{ volts} \rightarrow [-1 \text{ to } +1] \text{ normalized} * 10$$

In your block diagram, you can do this as follows.

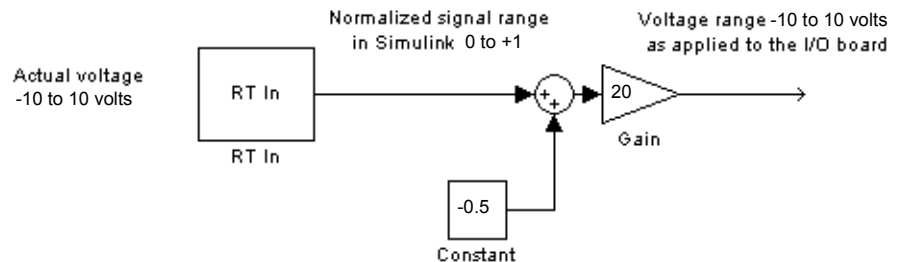


Choose -10 to +10 Volts and Normalized Unipolar

From the **Input range** list, choose -10 to +10 V, and from the **Block output signal** list, choose Normalized unipolar. This example converts a normalized bipolar value to volts, but you could also easily convert directly to another parameter in your model.

$$-10 \text{ to } 10 \text{ volts} \rightarrow ([0 \text{ to } 1] \text{ normalized} - 0.5) * 20$$

In your block diagram, do this as follows.



Normalized Scaling for Analog Outputs

Analog outputs are treated in an equivalent manner to analog inputs.

If the voltage range on the D/A converter is set to 0 to +5 volts, and the **Block input signal** is chosen as Normalized bipolar, then a Simulink signal of amplitude -1 results in an output voltage of 0 volts. Similarly, a Simulink signal of amplitude +1 results in an output voltage of +5 volts.

A voltage range on the D/A converter is set to -10 to +10 volts, and the **Block input signal** is chosen as Normalized bipolar, then a Simulink signal of amplitude -1 results in an output voltage of -10 volts. Similarly, a Simulink signal of amplitude +1 results in an output voltage of +10 volts.

This may require that you adjust your signal amplitudes in Simulink using a Gain block, Constant block, and Summer block depending on the selected voltage range.

Troubleshooting

Plots Not Visible in Simulink Scope Block	5-2
Compiler Error Message	5-3
Failure to Connect to Target	5-3
Sample Time Too Fast	5-4
S-Functions Using Math Functions	5-5

Solutions have been worked out for some common errors and problems that can occur when using the Real-Time Windows Target.

This chapter includes the following topics:

- “Plots Not Visible in Simulink Scope Block” on page 5-2
- “Compiler Error Message” on page 5-3
- “Failure to Connect to Target” on page 5-3
- “Sample Time Too Fast” on page 5-4

Plots Not Visible in Simulink Scope Block

For data to plot correctly in a Simulink Scope block, you must specify the following:

- `rtwinext` as the **MEX-File for external interface** on the **External Target Interface** dialog box
- **External** selected from the **Simulation** menu
- **Connect to target** selected from the **Simulation** menu
- Select one or more signals for capture (designated with “X”) in the **External Signal & Triggering** dialog box
- **Duration * Fixed Step Size** close to or less than the X range in the Scope block
- Correct mode (one-shot vs. normal)
- Appropriate signal levels to allow triggering
- Y range on Simulink Scope block axes large enough to span the signal amplitude
- X range
- **Arm when connect to target** in the **External Data Logging Configuration** dialog box or **arm** in the **External Data Logging Control Panel**
- **Start real-time code** selected from the **Simulation** menu

If you are unable to see signals plotted in your Simulink Scope blocks after all of the above items have been selected, failure to obtain time responses in Scope blocks may be due to insufficient CPU time. To determine CPU utilization, type `rtwho`. The `rtwho` command returns information about MATLAB performance. The value returned is an indicator of how much loading your model places on the CPU. If Scope blocks fail to plot, this may be an indication that insufficient

time is available between sample intervals to allow data to be transferred back to the MATLAB environment where the plotting is performed. To test for this condition, you can run one of the demonstration models, or you can try running your model at a significantly slower rate to determine whether this is the cause. We recommend that MATLAB performance does not fall below 80%.

Compiler Error Message

Possible problem — During the build phase of your model, the **Simulation Errors** dialog box displays

```
incorrect compiler installation
```

Solution — During the installation of the Watcom 10.6 or 11.0 compiler, select the **DOS target** check box in addition to selecting the **Windows target** check box.

Failure to Connect to Target

Possible Problem — When trying to connect to the target, the **Simulation Errors** dialog box displays

```
Checksum mismatch. Target code needs to be rebuilt
```

Solution — This indicates that the model structure has changed since the last time code was generated. You must rebuild the real-time application. If your model fails to successfully build, we recommend that you delete `.mk` and `.obj` files from the Real-Time Workshop project directory, and then select **Build** from the **Tools** menu.

Possible Problem — When trying to connect to the target, the Simulink diagnostic dialog box displays

```
External mode MEX-file "win_tgt" does not exist or is not on the  
MATLAB path
```

Solution — The Real-Time Windows Target Versions 1.0 and 1.5 used the MEX-file `win_tgt`. For the Real-Time Windows Target Version 2.2, the MEX-file name was changed to `rtwintext`. If you create a new Simulink model, the new filename is entered correctly. If you have Simulink models where you used the Real-Time Windows Target 1.0 or 1.5, you need to change the filename using the following procedure:

- 1 In the Simulink window, and from the **Tools** menu, click **External mode control panel**.
- 2 On the **External Mode Control Panel** dialog box, click the **Target interface** button.
- 3 In the **External Target Interface** dialog box, and in the **MEX-file for external mode** text box, enter

```
rtwinext
```
- 4 Click **Ok**.

Sample Time Too Fast

During a run, you may not see any output in the Scope window. This could indicate that the sample time is too small. In the MATLAB command window, type

```
rtwho
```

Check the value for MATLAB performance. A value less than 80% indicates that your sample time may be too small.

In general, we recommend that you start by choosing a slow sample rate. For example, select a sample time of 0.01 second, and confirm your system runs correctly and plots are displayed. Should you select a sample rate that exceeds the capability of your computer, an error message is displayed and real-time execution is terminated. If this occurs, select a slower sample rate. Then rebuild the model, connect to the target, and start the real-time application again. You must rebuild the real-time application after changing the sample time.

Check the MATLAB performance value returned when typing `rtwho`. If MATLAB performance is in the range of 98% or so, then consider decreasing your sample time by one order of magnitude.

If you notice either slow updates of Scope blocks or a complete failure to plot data in the Scope blocks, you may be reaching the upper threshold for the sample rate on your hardware. Plotting data has a lower priority than execution of your real-time application.

S-Functions Using Math Functions

Possible problem — When creating your own S-functions that include math functions, the S-functions compile okay, but you cannot build the application.

Solution — Add the Real-Time Windows Target header to your S-function. For example, add

```
#include<math.h>
#include"rtwintgt.h"
```

The header `#include<math.h>` must precede the header `#include"rtwintgt.h"`.

Supported I/O Boards

ISA Bus	A-2
PCMCIA Bus	A-11
PCI Bus	A-12
Compact PCI	A-14
PXI Bus	A-14
PC/104 Bus	A-15
Standard Devices	A-16

The Real-Time Windows Target includes support for more than 80 I/O boards. Multiple boards may be used as I/O for a model provided they have nonoverlapping base addresses. If you have a board that is not listed here, you can support it by adding your own I/O driver.

This appendix includes the following topics:

- “ISA Bus” on page A-2
- “PCMCIA Bus” on page A-11
- “PCI Bus” on page A-12
- “Compact PCI” on page A-14
- “PXI Bus” on page A-14
- “PC/104 Bus” on page A-15
- “Standard Devices” on page A-16

Note Some of the functions on a board may not be supported by the Real-Time Windows Target. Check the MathWorks Web site for an updated list of supported boards and functions at <http://www.mathworks.com/products/rtwt/ioboards.shtml>.

ISA Bus

This table lists the ISA bus I/O boards supported by the Real-Time Windows Target.

Table A-1: ISA Bus Supported I/O Boards

Manufacturer	Board Name
Advantech	PCL-1800
	PCL-711B
	PCL-712

Table A-1: ISA Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	PCL-714
	PCL-722
	PCL-724
	PCL-725
	PCL-726
	PCL-727
	PCL-728
	PCL-730
	PCL-731
	PCL-733
	PCL-734
	PCL-735
	PCL-812
	PCL-812PG
	PCL-814B
	PCL-816
	PCL-818
	PCL-818H
	PCL-818HD
	PCL-818HG
	PCL-818L
Analog Devices	RTI-800

Table A-1: ISA Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	RTI-800-A
	RTI-800-F
	RTI-815
	RTI-815-A
	RTI-815-F
Axiom	AX5032IO
	AX5064I
	AX50640
	AX5215H
	AX5244H
	AX5412-H
	AX5412-L
	AX5611C-H
	AX5611C-L

Table A-1: ISA Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
Computer Boards	CIO-DAC02
	CIO-DAC02/16
	CIO-DAC08
	CIO-DAC08/16
	CIO-DAC16
	CIO-DAC16/16
	CIO-DAS08
	CIO-DAS08-AOH
	CIO-DAS08-AOL
	CIO-DAS08-AOM
	CIO-DAS08/Jr

Table A-1: ISA Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	CIO-DAS08/Jr-AO
	CIO-DAS08/Jr/16
	CIO-DAS08/Jr/16-AO
	CIO-DAS08-PGH
	CIO-DAS08-PGL
	CIO-DAS08-PGM
	CIO-DAS-1401/12
	CIO-DAS-1402/12
	CIO-DAS-1402/16
	CIO-DAS16/330
	CIO-DAS16/F
	CIO-DAS16/Jr
	CIO-DAS16Jr/16
	CIO-DAS-1601/12
	CIO-DAS-1602/12
	CIO-DAS-1602/16
	CIO-DAS48-PGA

Table A-1: ISA Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	CIO-DDA06
	CIO-DDA06/JR
	CIO-DDA06/16
	CIO-DDA06/JR/16
	CIO-DI48
	CIO-DI96
	CIO-DI192
	CIO-DIO24
	CIO-DIO24H
	CIO-DIO48
	CIO-DIO48H
	CIO-DIO96
	CIO-DIO192
	CIO-DISO48
	CIO-DO24DD
	CIO-DO48DD
	CIO-DO48H
	CIO-DO96H
	CIO-DO192H
	CIO-DUAL-AC5
	CIO-PDISO8
	CIO-PDISO16

Table A-1: ISA Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	CIO-QUAD02
	CIO-QUAD04
	CIO-RELAY08
	CIO-RELAY16
	CIO-RELAY24
	CIO-RELAY32
Data Translation	DT2801
	DT2801-A
	DT2801/5716
	DT2805
	DT2805/5716
	DT-2809
	DT2811-PGH
	DT2811-PGL
	DT2820
Humusoft	AD512
	MC101-CE150
	MF604
Intelligent Instrumentation	PCI-20377W
	MF604
Keithley-Metrabyte	ADC-16
	DAC-02

Table A-1: ISA Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	DAS-1201
	DAS-1202
	DAS-1401
	DAS-1402
	DAS-1601
	DAS-1602
	DAS-16G1
	DAS-16G2
	DAS-8
	DAS-8/AO
	DAS-8PGA
	DAS-8PGA-G2
	DDA-06
	PDISO-8
	PIO-12
	PIO-24
	PIO-32IN
	PIO-32IO
	PIO-32OUT
	PIO-96
	PIO-HV
	REL-16

Table A-1: ISA Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
National Instruments	AT-MIO-16E-1
	AT-MIO-16E-2
	AT-MIO-64E-3
	AT-MIO-16E-10
	AT-MIO-16DE-10
	AT-AI-16XE-10
	AT-MIO-16XE-10
	AT-MIO-16XE-50
	Lab-PC
	Lab-PC+
	Lab-PC-1200
	Lab-PC-1200-AI
	PC-DIO-24
	PC-DIO-24PnP
PC-OPDIO-16	
Quanser	MultiQ-3
	MultiQ-PCI
Scientific Solutions	LabMaster DMA-PGH
	LabMaster DMA-PGL
Sensoray	Model 626
Technology80	Model 5312B

PCMCIA Bus

This table lists the PCMCIA bus I/O boards supported by the Real-Time Windows Target.

Table A-2: PCMCIA Bus Supported I/O Boards

Manufacturer	Board Name
Computer Boards	PC-CARD-DAS16/12
	PC-CARD-DAS16/16
	PC-CARD-DAS16/330
	PC-CARD-DIO48
	PCM-DAC08
	PCM-DAS08
	PCM-DAS16S/330
	PCM-DAS16D/12
	PCM-DAS16S/12
	PCM-DAS16D/16
PCM-DAS16S/16	
National Instruments	DAQCard-700
	DAQCard-1200
	DAQCard-6024E
	DAQCard-6062E
	DAQCard-AI-16E-4
	DAQCard-AI-16XE-50
DAQCard-DIO-24	

PCI Bus

This table lists the PCI bus I/O boards supported by the Real-Time Windows Target.

Table A-3: PCI Bus Supported I/O Boards

Manufacturer	Board Name
Advantech	PCI-1710
	PCI-1710HG
	PCI-1711
	PCI-1713
	PCI-1720
	PCI-1731
	PCI-1750
	PCI-1751
	PCI-1752
	PCI-1753
	PCI-1753E
	PCI-1754
	PCI-1756
Computer Boards	PCI-DAS08
	PCI-DAS1000
	PCI-DAS1001
	PCI-DAS1002
	PCI-DAS1200
	PCI-DAS1200/Jr

Table A-3: PCI Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	PCI-DAS1602/12
	PCI-DAS1602/16
	PCI-DAS1602/16/Jr
	PCI-DIO24
	PCI-DIO24H
	PCI-DIO48H
	PCI-DIO96
	PCI-DIO96H
	PCI-DUAL-AC5
	PCIM-DDA06/16
	PCI-QUAD04
National Instruments	PCI-1200
	PCI-6023E
	PCI-6024E
	PCI-6025E
	PCI-6031E
	PCI-6032E
	PCI-6033E
	PCI-6034E
	PCI-6035E
	PCI-6052E
	PCI-6071E

Table A-3: PCI Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	PCI-MIO-16E-1
	PCI-MIO-16E-4
	PCI-MIO-16XE-10
	PCI-MIO-16XE-50

Compact PCI

This table lists the compact PCI I/O boards supported by the Real-Time Windows Target.

Table A-4: Compact PCI Supported I/O Boards

Manufacturer	Board Name
Computer Boards	CPCI-DIO24H
	CPCI-DIO48H
	CPCI-DIO96H

PXI Bus

This table lists the PXI bus I/O boards supported by the Real-Time Windows Target.

Table A-5: PXI Bus Supported I/O Boards

Manufacturer	Board Name
National Instruments	PXI-6025E
	PXI-6030E

Table A-5: PXI Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	PXI-6031E
	PXI-6035E
	PXI-6040E
	PXI-6052E
	PXI-6070E
	PXI-6071E

PC/104 Bus

This table lists PC/104 bus I/O devices supported by the Real-Time Windows Target.

Table A-6: PC/104 Bus Supported I/O Boards

Manufacturer	Board Name
Advantech	PCM-3718H
	PCM-3718HG
	PCM-3724
	PCM-3725
	PCM-3730
Computer Boards	PC104-AC5
	PC104-DAC06
	PC104-DAS08
	PC104-DAS16Jr/12
	PC104-DAS16Jr/16

Table A-6: PC/104 Bus Supported I/O Boards (Continued)

Manufacturer	Board Name
	PC104-DIO48
	PC104-DO48H
	PC104-PDISO8

Standard Devices

This table lists standard I/O devices supported by the Real-Time Windows Target.

Table A-7: Standard Supported I/O Devices

Manufacturer	Board Name
Other	Generic 8-bit Port
	Generic 16-bit Port
	Generic 32-bit Port
	Generic I8255
	Microsoft Windows Joystick
	Microsoft Windows Mouse
	Microsoft Windows Parallel Port (LPT)

Custom I/O Driver Blocks

Source Code for DOS Target Drivers	B-2
Incompatibility with Win32 API Calls	B-3
Nonsupported C Functions	B-3
Supported C Functions	B-4

Custom I/O device drivers can be used in combination with the Real-Time Windows Target. Due to the additional complexity of device drivers supplied with the Real-Time Windows Target, we recommend that device drivers be written in the same style as the device drivers that are provided with the DOS target support included with Real-Time Workshop.

We do not recommend using Analog Input, Analog Output, Digital Input, or Digital Output drivers as a starting point for creating custom device drivers.

This appendix includes the following topics:

- “Source Code for DOS Target Drivers” on page B-2
- “Incompatibility with Win32 API Calls” on page B-3
- “Nonsupported C Functions” on page B-3
- “Supported C Functions” on page B-4

Source Code for DOS Target Drivers

Source code for the DOS target device drivers is located in `matlabroot\rtw\c\dos\devices`. This table lists the available DOS device drivers.

Table B-1: DOS Device Drivers Included with the Real-Time Workshop

Type of Device Driver	Filename
Analog to Digital	das16ad.c das16ad.h das16ad.tlc
Digital to Analog	das16da.c das16da.h das16da.tlc
Digital Input	das16di.c das16di.h das16di.tlc
Digital Output	das16do.c das16do.h das16do.tlc

Device drivers written in this style (that is, as inlined S-functions) are compatible with the Real-Time Windows Target. They may be used in combination with the device drivers provided with the Real-Time Windows Target.

See the Real-Time Workshop documentation for more information about these I/O device drivers and writing custom device drivers.

Incompatibility with Win32 API Calls

The Real-Time Windows Target kernel intercepts the interrupt from the system clock. It then reprograms the system clock to operate at a higher frequency for running your real-time application. At the original clock frequency, it sends an interrupt to the Windows operating system to allow Windows applications or any software using the Win32 API to run.

As a result, software that uses the Win32 API may not be executed as a component of your real-time application. Any software you use to write I/O drivers must not have any calls to the Win32 API.

Nonsupported C Functions

If you create your own custom I/O driver blocks, you should first check for C functions that are supported by the Real-Time Windows Target.

Functions that use the Windows operating system are not supported with the Real-Time Windows Target. This is because the kernel intercepts the system clock and first runs the real-time application. If there is time left before the next sample time, the kernel may allow a Windows application or function to run.

The following list includes many, but not all of the nonsupported functions:

- **File I/O** — fopen, freopen, fclose, fread, fwrite, fputs, fputc, fgets, fgetc, gets, getc, getchar, puts, putc, putchar, fflush, setbuf, setvbuf
- **Console I/O** — printf, fprintf, sprintf, vsprintf, vprintf, vsprintf, fscanf, scanf, sscanf
- **Process management** — spawn, exit, abort, atexit
- **Signals and exceptions** — signal, longimp, raise

- **Time functions** — clock, time, difftime, asctime, ctime, difftime, gmtime, localtime, mktime, strftime
- **Win32 API functions.** No Windows API functions are supported.

Supported C Functions

You can use ANSI C functions that do not use the Windows operating system in your custom blocks or I/O drivers. The following includes a partial list of supported functions:

- **Data conversion functions** — abs, atof, atoi, atol, itoa, labs, ltoa, strtod, strtol, strtoul, ultoa
- **Memory allocation functions** — calloc, free, malloc
- **Memory manipulation functions** — _memccpy, memcpy, memchr, memcmp, _memicmp, memmove, memset
- **String manipulation functions** — strcat, strchr, strcmp, strcpy, strcspn, _strdup, _stricmp, strlen, _strlwr, strcat, strncmp, strncpy, _strnset, strpbrk, strrchr, _strev, _strset, strspn, strstr, strtok,strupr
- **Mathematical functions** — acos, asin, atan, atan2, ceil, cos, cosh, div, exp, fabs, floor, fmod, frexp, ldexp, ldiv, log, log10, max, min, modf, pow, rand, sin, sinh, sqrt, srand, tan, tanh, uldiv
- **Character class tests and conversion** — isalnum, isalpha, _isascii, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit, isxupper, isxlower, _toascii, tolower, toupper
- **Searching and sorting** — bsearch, qsort
- **Dummy functions** — exit, fprintf, printf

A

A/D

See analog-to-digital

adding

Analog Input block 4-8

I/O driver blocks 4-8

input blocks 4-8

analog input

normalized scaling 4-33

Analog Input block

configuring 4-12

Analog Output block

configuring 4-14

analog-to-digital

channel selection 4-29

application

Real-Time Windows Target 1-5

B

before you install

obtaining a valid license 2-5

build process xv

C

C compiler

required product xi

third party 2-12

capturing and displaying signals 1-3

CD

installation 2-6

changing parameters

parameter tuning 1-7

channel selection

entering configurations 4-29

compact PCI

installing 4-6

compact PCI bus

supported boards A-14

compatibility

with MATLAB ix

with Real-Time Workshop xi

with Simulink x

with the MathWorks software 1-4

compiler

error message 5-3

Microsoft Visual C/C++ xi

required xi

selecting 2-12

third party 2-12

Watcom C/C++ 2-12

computer

PC compatible 1-8

configuring

Analog Input block 4-12

Analog Output block 4-14

Counter Input block 4-22

Digital Input block 4-17

Digital Output block 4-19

Encoder Input block 4-25

I/O boards 4-3

connecting

real-time application 3-21

Simulink model 3-21

contacting The MathWorks

for valid license 2-5

conventions

in this guide xv

typographical xv

Counter Input block

configuring 4-22

- creating
 - real-time application 3-20
 - Simulink model 1-11
- custom I/O drivers
 - creating B-2
 - DOS target drivers B-2
 - incompatible with Win32 B-3
 - overview B-2
 - source code B-2

D

D/A

See digital-to-analog

- data archiving parameters
 - entering 3-43
 - logging data to disk drive 3-43
- data buffers 1-13
- definitions
 - terms xv
- demo library
 - opening 2-22
 - Real-Time Windows Target 2-22
- Demux block
 - separating I/O signals 4-27
- description
 - Simulink external mode 1-12
- device drivers
 - channel selection 4-29
 - custom I/O 4-36
 - writing custom B-2
- Digital Input block
 - configuring 4-17
- Digital Output block
 - configuring 4-19
- digital-to-analog
 - channel selection 4-29

- directories
 - installed 2-9
 - MATLAB working 2-9
 - Real-Time Workshop working 2-9
 - working 2-9
 - xpc 2-9
 - xpcdemo 2-9
- disk drive
 - plotting logged data 3-46
 - signal logging 3-37
- documentation
 - notational conventions xv
 - terminology conventions xv
- DOS target drivers
 - use for custom drivers B-2
- DSP Blockset
 - compatible software 1-4

E

- education
 - control engineers 1-3
 - signal processing engineers 1-3
- Encoder Input block
 - configuring 4-25
- entering
 - data archiving parameters 3-43
 - scope properties 3-37
 - signal and triggering properties 3-40
 - simulation parameters for Real-Time Workshop 3-14
 - simulation parameters for Simulink 3-7
- error message
 - with compiler 5-3
- execution
 - real-time 1-10
 - running in real time 1-11

- expected background
 - experienced users xiii
 - new users xiii
- external interface MEX-file
 - rtwinext 5-3
 - win_tgt 5-3
- external mode
 - description 1-12
 - parameter tuning 1-7
- F**
- failure to connect
 - troubleshooting 5-3
- features
 - signal logging 1-6
 - signal tracing 1-6
- files
 - application 2-9
 - external mode interface 2-9
 - I/O drivers 2-9
 - installed 2-9
 - kernel install command 2-9
 - make command 2-9
 - makefile 2-9
 - project directory 2-9
 - system target 2-9
 - system target file 2-10
 - template makefile 2-10
 - working directory 2-9
 - xpc directory 2-9
 - xpcdemos directory 2-9
- Fixed-Point Blockset
 - compatible software 1-4

- H**
- hardware
 - requirements 2-3
 - system requirements 2-3

- I**
- I/O blocks
 - Analog Input block 4-12
 - Analog Output block 4-14
 - Counter Input block 4-22
 - Digital Input block 4-17
 - Digital Output block 4-19
 - Encoder Input block 4-25
 - input and output 4-8
 - separating signals 4-27
 - with Simulink x
- I/O boards
 - compact PCI boards 4-6
 - configuring 4-3
 - installing 4-3
 - ISA bus 4-5
 - list of supported A-2
 - overview 4-3
 - PC/104 bus 4-6
 - PCI bus 4-6
 - PCMCIA 4-6
- I/O drivers
 - characteristics 4-32
 - custom B-2
 - using 4-32
- input blocks
 - adding Analog Input blocks 4-8
 - overview 4-8
- input/output
 - support 1-8

installation
 CD 2-6
 from Web downloadable 2-7

installing
 compact PCI 4-6
 I/O boards 4-3
 kernel overview 2-14
 Real-Time Windows Target 2-5
 testing installation 2-18

ISA bus
 installing 4-5

ISA-bus
 supported boards A-2

J

joystick
 supported device A-16

K

kernel
 communication with hardware 1-4
 installing 2-14
 scheduler 1-4
 timer interrupt 1-4
 uninstalling 2-16

L

licenses
 getting or updating 2-5
 system requirements 2-5

list
 supported I/O boards A-2

logging
 data to disk drive 3-43

data to workspace 3-29

M

makefile 2-10

MathWorks
 compatible software 1-4
 DSP Blockset 1-4
 Fixed-Point Blockset 1-4
 Simulink blocks 1-4
 Stateflow/Stateflow Coder 1-4

MATLAB

 compatibility ix
 required product ix

MATLAB workspace
 signal logging 3-29

MEX utility
 selecting compiler 2-12

Microsoft

 Visual C/C++ compiler xi

Microsoft Visual C/C++
 compatible versions xi

model parameters
 changing 3-49
 parameter tuning 3-49

mouse
 supported device A-16

N

nonreal-time
 simulation 3-12

normalized scaling
 analog input 4-33

O

- opening
 - demo library 2-22
- organization
 - of this document xiv
- output blocks
 - overview 4-8
- overview
 - custom I/O drivers B-2
 - I/O boards 4-3
 - input blocks 4-8
 - installing kernel 2-14
 - output blocks 4-8
 - parameter tuning 3-48
 - real-time application 3-14
 - Real-Time Windows Target 1-3
 - system concepts 1-12
 - system requirements 2-3
 - testing installation 2-18

P

- parallel port
 - supported device A-16
- parameter tuning
 - changing model parameters 3-49
 - changing parameters 1-7
 - external mode 1-7
 - feature 1-7
 - overview 3-48
- PCI bus
 - installing 4-6
- PCI-bus
 - supported boards A-12
- PCMCIA bus
 - installing 4-6
 - supported boards A-11

- plots not visible
 - troubleshooting 5-2
- plotting
 - logged data from disk 3-46
 - logged data from workspace 3-35

R

- real-time
 - control 1-3
 - execution 1-10
 - hardware-in-the-loop 1-3
 - signal processing 1-3
- real-time application
 - and the development process 1-11
 - connecting to Simulink model 3-21
 - creating 3-20
 - overview 3-14
 - Real-Time Windows Target 1-5
 - Real-Time Workshop parameters 3-14
 - scope properties for signal tracing 3-17
 - simulation parameters for Real-Time Workshop 3-14
 - software environment 1-5
 - starting 3-25
 - stopping 3-25
- real-time kernel
 - Real-Time Windows Target 1-4
 - scheduler 1-4
 - software environment 1-4
 - timer interrupt 1-4

Real-Time Windows Target

- application 1-5
 - custom I/O device drivers 4-36
 - demo library 2-22
 - development process 1-11
 - files 2-9
 - I/O board support A-2
 - installing kernel 2-14
 - overview 1-3
 - overview of install 2-5
 - real-time application 1-5
 - real-time kernel 1-4
 - required products ix
 - software environment 1-10
 - uninstalling kernel 2-16
 - what is it? 1-3
- Real-Time Workshop**
- compatibility xi
 - entering simulation parameters 3-14
 - required product xi
- required products**
- C language compiler xi
 - MATLAB ix
 - overview ix
 - Real-Time Workshop xi
 - Simulink x
- requirements**
- hardware 2-3
 - software 2-4
- rtvdp.mdl**
- Simulink model 2-18
- rtwinext**
- external interface MEX-file 5-3
- running**
- execution in real time 1-11
 - real-time application 3-25
 - simulation in nonreal time 3-12

S

- sample rates
 - excessive 2-22
- sample time
 - too fast 5-4
- scope properties
 - entering 3-37
 - entering for signal tracing 3-17
 - for signal logging to disk drive 3-37
 - for signal logging to workspace 3-29
- selecting
 - C compiler 2-12
- separating
 - I/O signals 4-27
- setting
 - initial working directory 2-11
 - working directory 2-11
- signal and triggering
 - entering properties 3-31
 - properties 3-40
- signal archiving
 - See* signal logging
- signal data
 - plotting from disk drive 3-46
 - plotting from workspace 3-35
- signal logging
 - entering scope properties 3-29
 - feature 1-6
 - plotting data 3-35
 - signal and triggering properties 3-40
 - to disk drive 3-37
 - to MATLAB workspace 3-29
- signal logging to disk drive
 - data archiving parameters 3-43
 - signal and triggering properties 3-40

- signal logging to workspace
 - scope properties 3-29
 - signal and triggering properties 3-31
 - signal processing
 - DSP Blockset 1-4
 - signal tracing
 - feature 1-6
 - scope properties 3-17
 - signals
 - capturing and displaying 1-3
 - simulation
 - nonreal-time 3-12
 - running in nonreal time 1-11
 - simulation parameters
 - entering 3-7
 - for Real-Time Workshop 3-14
 - Simulink
 - compatibility x
 - compatible software 1-4
 - required product x
 - running a simulation 3-12
 - Simulink external mode
 - description 1-12
 - parameter tuning 1-7
 - Simulink model
 - and the development process 1-11
 - connect to real-time application 3-21
 - creating 1-11
 - rtvdp.mdl 2-18
 - software
 - system requirements 2-4
 - software environment
 - overview 1-10
 - real-time application 1-5
 - real-time kernel 1-4
 - requirements 2-4
 - standard devices
 - joystick A-16
 - mouse A-16
 - parallel port A-16
 - starting
 - real-time application 3-25
 - Stateflow
 - compatible software 1-4
 - stopping
 - real-time application 3-25
 - support
 - input/output 1-8
 - supported I/O boards
 - compact PCI bus A-14
 - ISA bus A-2
 - list A-2
 - PC/104 bus A-15
 - PCI bus A-12
 - PCMCIA bus A-11
 - PXI bus A-14
 - system concepts
 - data buffers 1-13
 - overview 1-12
 - transferring data 1-13
 - system requirements
 - hardware 2-3
 - licenses 2-5
 - overview 2-3
 - software 2-4
 - software environment 2-4
 - updating licenses 2-5
 - system target file 2-10
- T**
- template makefile 2-10
 - terms
 - definitions xv

testing installation

overview 2-18

third-party

C compiler 2-12

transferring data 1-13

troubleshooting

compiler error message 5-3

failure to connect 5-3

incorrect MEX-file 5-3

plots not visible 5-2

sample time too fast 5-4

typographical conventions (table) xvii

U

uninstalling

kernel 2-16

using

I/O device drivers 4-32

using this guide

conventions xv

organization xiv

V

valid license

obtaining 2-5

Visual C/C++

compatible versions xi

compiler 2-12

W

Watcom

C/C++ compiler 2-12

Web downloadable

installing from 2-7

Win32

incompatible with I/O drivers B-3

Windows Target

See Real-Time Windows Target

working directory

initial 2-11

setting 2-11

setting initial 2-11

writing customized device drivers B-2